BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 08-139935
(43)Date of publication of application : 31.05.1996

(51)Int.Cl. H04N 1/41
H03H 17/02
H03M 7/40
H04N 7/30

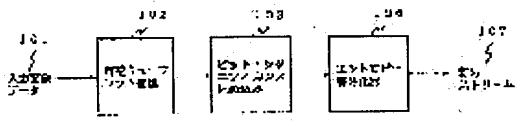
(21)Application number : 07-237987
(22)Date of filing : 18.09.1995

(71)Applicant : RICOH CO LTD
(72)Inventor : AAMADO ZANDEI
JIEIMUSU DEII AREN
EDOWAADO ERU SHIYUWARU
MAATEIN PII BOORITSUKU

(30)Priority
Priority number : 94 310146 Priority date : 21.09.1994 Priority country : US

(54) CODER AND WAVELET CONVERSION FILTER

(57)Abstract:
PURPOSE: To provide an efficient coder employing conversion to obtain excellent energy concentration and the wavelet conversion filter for the purpose.
CONSTITUTION: A reversible wavelet conversion block 102 divides image data 101 by using a non minimum length reversible filter to convert divisions into a code series. A bit significance imbeded block 103 converts the coefficient series into a form of code-absolute value and applies a frequency base model or an integrated space/frequency base model to the converted series. The obtained decision is subjected to entropy coding by an entropy coder 104.



LEGAL STATUS

[Date of request for examination] 10.04.2001

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]	3436832
[Date of registration]	06.06.2003
[Number of appeal against examiner's decision of rejection]	
[Date of requesting appeal against examiner's decision of rejection]	
[Date of extinction of right]	

登録（2004/05/25）

4199603000961399935

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（A）　(11)特許出願公開番号

特開平8－139935

(43)公開日　平成8年(1996)5月31日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｈ０４Ｎ　1/41	Ｂ			
Ｈ０３Ｈ　17/02	Ｂ	8842－5Ｊ		
Ｈ０３Ｍ　7/40		9382－5Ｋ		
Ｈ０４Ｎ　7/30				
			Ｈ０４Ｎ　7/133	Ｚ

審査請求　未請求　請求項の数24　ＯＬ　（全71頁）

(21)出願番号　特願平7－237987

(22)出願日　平成7年(1995)9月18日

(31)優先権主張番号　０８／３１０１４６

(32)優先日　1994年9月21日

(33)優先権主張国　米国（ＵＳ）

(71)出願人　000006747
株式会社リコー
東京都大田区中馬込１丁目３番６号

(72)発明者　アーマド・ザンディ
アメリカ合衆国　カリフォルニア州
94025　メンローパーク　サンド　ヒル
ロード　2882　リコーコーポレーション内

(72)発明者　ジェイムス　ディー　アレン
アメリカ合衆国　カリフォルニア州
94025　メンローパーク　サンド　ヒル
ロード　2882　リコーコーポレーション内

(74)代理人　弁理士　鈴木　誠　（外１名）

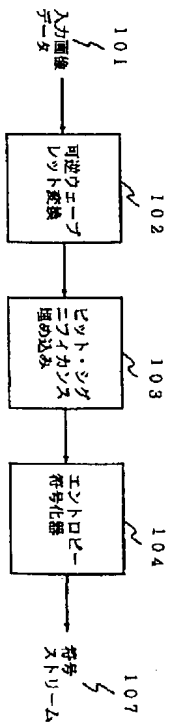
最終頁に続く

(54)【発明の名称】　符号化装置及びウエーブレット変換フィルタ

(57)【要約】

【課題】　良好なエネルギー集中を得られる変換を用いた効率的な符号化装置と、そのためのウエーブレット変換フィルタを提供する。

【解決手段】　可逆ウエーブレット変換ブロック１０２は画像データ１０１を非最小長可逆フィルタを用いて分割し符号系列に変換する。ビット・シグニフィカンス埋め込みブロック１０３は、その係数系列を符号－絶対値形式に変換した後に周波数ベースモデルまたは統合空間／周波数ベースモデルを適用する。得られたデシジョンをエントロピー符号化器１０４でエントロピー符号化する。



1

【特許請求の範囲】

【請求項１】　入力データを圧縮データストリームに符号化するための符号化装置であって、

該入力データを複数の係数に変換する可逆ウエーブレットフィルタ、

該複数の係数に対し埋め込み符号化を行なうことにより、ビットストリームを生成する、該可逆ウエーブレットフィルタに接続された埋め込み符号化器、及び該ビットストリームに対しエントロピー符号化を行なって、符号化データを生成する、該埋め込み符号化器に接続されたエントロピー符号化器からなる符号化装置。

【請求項２】　入力データを符号化するための符号化装置であって、

該入力データを受け取って、該入力データの分割したものを表わす係数の系列を生成する変換符号化器、及び該係数の系列を受け取り、該係数の系列に対しビット・シグニフィカンス符号化を行なって符号化データを生成する埋め込み符号化器からなり、該埋め込み符号化器が該係数の系列の全部を受け取る前に符号化データを発生する符号化装置。

【請求項３】　請求項２記載の符号化装置において、該変換符号化器及び該埋め込み符号化器は、１パスで該入力データから符号化データを生成するように動作することを特徴とする符号化装置。

【請求項４】　入力データ信号のフィルタリングのためのウエーブレット変換フィルタであって、

該入力データ信号のサンプルの第１のペアを加算して第１の結果を生成する第１加算器、

該第１の結果に基づいた第１の低域通過係数を出力する第１出力論理、

ある低域通過係数と該第１低域通過係数の入力を有し、該ある低域通過係数を該第１低域通過係数から減算して第２の結果を発生する第１減算器、ここにおいて該ある低域通過係数は該第１減算器にフィードバックとして受け取られる、

該入力データ信号のサンプルの第２のペアを相互に減算して第３の結果を発生する第２減算器、

該第３結果に基づく入力と該第２結果を受け取って加算し第４の結果を発生する第２加算器、

該第４結果に基づいた第２の低域通過係数を生成する第２出力論理からなり、該第１及び第２の低域通過係数を出力するウエーブレット変換フィルタ。

【請求項５】　請求項４記載のウエーブレット変換フィルタにおいて、該第１出力論理は除算器からなることを特徴とするウエーブレット変換フィルタ。

【請求項６】　請求項５記載のウエーブレット変換フィルタにおいて、該除算器はビットシフタからなることを特徴とするウエーブレット変換フィルタ。

【請求項７】　請求項４記載のウエーブレット変換フィルタにおいて、該第２出力論理は除算器からなることを

2

特徴とするウエーブレット変換フィルタ。

【請求項８】　請求項７記載のウエーブレット変換フィルタにおいて、該除算器はビットシフタからなることを特徴とするウエーブレット変換フィルタ。

【請求項９】　入力データ信号のフィルタリングのためのウエーブレット変換フィルタであって、

該入力データ信号のサンプルの第１のペアを加算して第１の結果を生成する第１加算器、

該第１結果を受け取り、該第１結果に第１の因数を乗じて第２の結果を得る第１乗算器、ここにおいて該第２結果は第１の低域通過係数として出力される、

ある低域通過係数を受け取り該第２結果より減算して第３の結果を発生する第１減算器、ここにおいて、該ある低域通過係数は該第１減算器にフィルードバックとして受け取られる、

該入力データ信号のサンプルの第２のペアを相互に減算して第４の結果を発生する第２減算器、

該第４結果を受け取って該第４結果を第２の因数で除して第５の結果を発生する第１除算器、

該第３結果及び該第５結果を受け取って加算し第６の結果を発生する第２加算器、及び該第６結果を受け取り該第６結果を第３の因数で除して第２の低域通過係数を生成する第２除算器からなり、該第１低域通過係数及び該第２低域通過係数が出力されるウエーブレット変換フィルタ。

【請求項１０】　請求項９記載のウエーブレット変換フィルタにおいて、該乗算器はシフタからなるなることを特徴とするウエーブレット変換フィルタ。

【請求項１１】　請求項９記載のウエーブレット変換フィルタにおいて、該除算器のすくなくとも一つはシフタからなることを特徴とするウエーブレット変換フィルタ。

【請求項１２】　入力データ信号のフィルタリングのためのウエーブレット変換フィルタであって、

１対の低域通過係数の減算をして第１の結果を発生する第１減算器、

該第１結果に基づいた第１の入力を、ある高域通過係数から減算して第２の結果を発生する第２減算器、

該第２結果を、ある低域通過係数に基づいた第２入力に加算して第３の結果を発生する第１加算器、

該第２結果を、該ある低域通過係数から減算して第４の結果を発生する第３減算器からなり、該第３結果に基づいた第１のサンプルと該第４結果に基づいた第２のサンプルを出力するウエーブレット変換フィルタ。

【請求項１３】　該第３結果を受け取り、該第３結果をクリップして該第１サンプルを発生する第１クリップ機構をさらに具備することを特徴とする請求項１２記載のウエーブレット変換フィルタ。

【請求項１４】　該第１結果を受け取り、該第１結果を第１因数により除して該第１入力を発生する除算器をさ

らに具備することを特徴とする請求項１２記載のウエーブレット変換フィルタ。

【請求項１５】　該ある低域通過係数を受け取り、該ある低域通過係数を乗算して該第２入力を生成する乗算器をさらに具備することを特徴とする請求項１２記載のウエーブレット変換フィルタ。

【請求項１６】　複数のデータシンボルを符号化するための符号化装置であって、

該複数のデータシンボルを受け取って該複数のデータシンボルをフォーマットされたデータシンボルの集合にフォーマットするフォーマッティングユニット、

該複数のデータシンボルに応じて複数のデシジョンを生成する、該フォーマッティングユニットに接続されたシグニフィカンスユニット、及び該シグニフィカンスユニットから該複数のデシジョンを受け取って記憶または出力するメモリ機構からなる符号化装置。

【請求項１７】　請求項１６記載の符号化装置において、該複数のデータシンボルは複数の係数からなることを特徴とする符号化装置。

【請求項１８】　請求項１６記載の符号化装置において、該フォーマッティングユニットは、該複数のデータシンボルを符号－絶対値形式にフォーマットされたデータの集合へ変換するための符号－絶対値ユニットからなることを特徴とする符号化装置。

【請求項１９】　請求項１８記載の符号化装置において、該符号－絶対値ユニットは、

各データシンボルに対するシグニフィカンス表示を出力するシグニフィカンス判定機構、

該複数のデータシンボルのそれぞれを受け取り、そのデータシンボルに対する符号及び仮数を生成する符号及び仮数ジェネレータ、及び該複数のデータシンボルのそれぞれに対するインデックスを生成するインデックスカウンタからなることを特徴とする符号化装置。

【請求項２０】　請求項１９記載の符号化装置において、該シグニフィカンス判定機構はプライオリティエンコーダからなることを特徴とする符号化装置。

【請求項２１】　請求項１９記載の符号化装置において、該シグニフィカンス判定機構は、１対のプライオリティエンコーダと、該１対のプラオリティエンコーダのそれぞれの出力の間で選択することによって該シグニフィカンス表示を出力する選択機構とからなることを特徴とする符号化装置。

【請求項２２】　該複数のフォーマットされたデータシンボルを受け取って記憶する絶対値メモリをさらに具備することを特徴とする請求項１６記載の符号化装置。

【請求項２３】　請求項２２記載の符号化装置において、該複数のフォーマットされたデータシンボルのそれぞれは、シグニフィカンスレベルに基づいて該絶対値メモリに格納されることを特徴とする符号化装置。

【請求項２４】　請求項２２記載の符号化装置において、該絶対値メモリは複数の記憶領域を具備し、該複数の記憶領域のそれぞれは別個のシグニフィカンスレベルのためのデータを格納し、

該絶対値メモリは複数のカウンタをさらに具備し、該複数のカウンタのそれぞれは、該フォーマットされたデータシンボルの格納のためのアドレスを提供するため該複数の記憶領域の１つに関係付けられることを特徴とする符号化装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明はデータ圧縮及び伸長システムの分野に係り、特に、圧縮／伸長システムにおけるデータの非損失性（lossless）及び損失性（lossy）符号化及び復号のための方法及び装置に関する。

【０００２】

【従来の技術】データ圧縮は、大量のデータの蓄積及び伝送のために非常に有用なツールである。例えば、文書のファクシミリ伝送のような画像の伝送に要する時間は、圧縮を使ってその画像の再生に必要とされるビット数を減らすと、飛躍的に短縮される。

【０００３】従来より、多くの様々なデータ圧縮手法が存在している。圧縮手法は２つの大まかなカテゴリー、つまり損失性符号化と非損失性符号化とに分類できる。損失性符号化とは、情報の損失を生じ、したがってオリジナルデータの完全な再現が保証されない符号化のことである。損失性符号化の目標とするところは、オリジナルデータから変化しても、その変化が不快であったり目だったりしないようにすることである。非損失性圧縮では、全ての情報が保存され、データは完全な再現が可能な方法で圧縮される。

【０００４】非損失性圧縮では、入力シンボルもくしは輝度データが出力符号語に変換される。入力としては、画像データ、音声データ、１次元データ（例えば空間的または時間的に変化するデータ）、２次元データ（例えば２つの空間軸方向に変化する（または１つの空間次元と１つの時間次元で変化する）データ）、あるいは多次元／マルチスペクトルのデータがあろう。圧縮がうまくいけば、その符号語は、符号化されない入力シンボル（または輝度データ）のために必要とされるビット数より少ないビット数で表現される。

【０００５】非損失性符号化法には、辞書符号化方式（例えば、Lempel-Ziv方式）、ランレングス符号化方式、計数符号化方式、エントロピー符号化方式がある。非損失性画像圧縮では、圧縮は予測またはコンテキストと符号化に基づいている。ファクシミリ圧縮用のＪＢＩＧ規格と、連続濃淡画像用のＤＰＣＭ（differntialpulse code modulation－ＪＰＥＧ規格のオプション）は画像用非損失性圧縮の例である。

【０００６】損失性圧縮では、入力シンボルまは輝度データは、量子化されてから出力符号語へ変換される。量

5

子化は、データの重要な特徴量を保存する一方、重要でない特徴量を除去することを目的としている。損失性圧縮システムは、量子化に先立ち、エネルギー集中をするために変換を用いる。

【０００７】画像信号処理における最近の開発は、効率的かつ高精度のデータ圧縮符号化方式を追求することに関心を集中し続けている。変換またはピラミッド信号処理の様々な方式が提案されており、その中にマルチ解像度ピラミッド処理方式とウエーブレット（wavelet）ピラミッド処理方式とがある。これら２方式は、サブバンド処理方式及び階層処理方式とも呼ばれる。画像データのウエーブレット・ピラミッド処理方式は、直交ミラーフィルタ（ＱＭＦ）を用いてオリジナル画像のサブバンド分割をする特殊なマルチ解像度ピラミッド処理方式である。なお、他の非ＱＭＦウエーブレット方式が存在する。

【０００８】ウエーブレット処理方式に関し、より多くの情報を得るには、Antonini, M., et al., "Image Coding Using Wavelet Transform", IEEE Transactions onImage Processing, Vol.1,No.2, April 1992、及びShapiro, J., "An EmbeddedHierarchical Image Coder Using Zerotrees of Wavelet Coefficients", Proc. IEEE Data Compression Conference, pgs.214-223, 1993を参照されたい。

【０００９】従来のウエーブレット処理方式の多くの問題点は、データ全部をその処理中に記憶しておくために大きなメモリが必要となることである。換言すれば、ウエーブレット処理を実施する場合、データ全部を、その符号化がなされる前に調べなければならない。かかる場合、少なくとも１回、データ全部のフルパスを完了するまでは全くデータ出力がない。実際、ウエーブレット処理は普通、データのマルチパスを必要とする。それ故に、大きなメモリがしばしば必要になる。大きなメモリを必要とせずに、ウエーブレット処理を利用することが望ましい。さらに、データの単一パスだけでウエーブレット処理を遂行することが望ましい。

【００１０】多くのウエーブレットまたはサブバンド変換装置は特定の正規型のフィルタを必要とする。例えば、低域通過フィルタと高域通過フィルタは同一長でなければならず、その係数の２乗和は１でなければならず、高域通過フィルタは時間－周波数特性が低域通過フィルタと正反対でなければならない、等々である。（１９９１年５月にLawton等に発行された米国特許第５，０１４，１３４号を参照）。より広範な種類のフィルタを許容することが望まれる。すなわち、低域通過フィルタと高域通過フィルタは同一長でなく、その係数の２乗和は１である必要がなく、高域通過フィルタは低域通過フィルタと正反対の時間－周波数特性である必要がない等々、そのような低域通過フィルタと高域通過フィルタを使用するウエーブレットまたはサブバンド変換装置を提

6

供することが望まれる。

【００１１】

【発明が解決しようと課題】本発明は、前述の問題点に鑑み、効率的な非圧縮性／圧縮性データ圧縮／伸長システムを実現しようとするもので、より具体的には、良好なエネルギー集中を得られる変換を用いた効率的な符号化装置と、それに関連した効率的な圧縮を可能にするウエーブレット変換フィルタとを提供することを目的とする。

【００１２】

【課題を解決するための手段】本発明によれば、入力データを複数の係数に変換する可逆ウエーブレットフィルタと、該複数の係数に対し埋め込み符号化を行なうことにより、ビットストリームを生成する、該可逆ウエーブレットフィルタに接続された埋め込み符号化器と、該ビットストリームに対しエントロピー符号化を行なって、符号化データを生成する、該埋め込み符号化器に接続されたエントロピー符号化器からなる符号化装置が提供される。

【００１３】本発明によれば、入力データを受け取って、該入力データの分割したものを表わす係数の系列を生成する変換符号化器と、該係数の系列を受け取り、該係数の系列に対しビット・シグニフィカンス符号化を行なって符号化データを生成する埋め込み符号化器とからなり、該埋め込み符号化器が該係数の系列の全部を受け取る前に符号化データを発生する符号化装置が提供される。この符号化装置は例えば、該変換符号化器及び該埋め込み符号化器が１パスで該入力データから符号化データを生成するように動作するように構成される。

【００１４】本発明によれば、入力データ信号のサンプルの第１のペアを加算して第１の結果を生成する第１加算器と、該第１の結果に基づいた第１の低域通過係数を出力する第１出力論理と、ある低域通過係数と該第１低域通過係数の入力を有し、該ある低域通過係数を該第１低域通過係数から減算して第２の結果を発生する第１減算器（該ある低域通過係数は該第１減算器にフィードバックとして受け取られる）と、該入力データ信号のサンプルの第２のペアを相互に減算して第３の結果を発生する第２減算器と、該第３結果に基づく入力と該第２結果を受け取って加算し第４の結果を発生する第２加算器と、該第４結果に基づいた第２の低域通過係数を生成する第２出力論理とからなり、該第１及び第２の低域通過係数を出力するウエーブレット変換フィルタが提供される。このウエーブレット変換フィルタにおいて、該第１出力論理及び／または該第２出力論理は、例えば除算器からなり、該除算器は例えばビットシフタからなる。

【００１５】本発明によれば、入力データ信号のサンプルの第１のペアを加算して第１の結果を生成する第１加算器と、該第１結果を受け取り、該第１結果に第１の因数を乗じて第２の結果を得る第１乗算器（該第２結果は

第１の低域通過係数として出力される）と、ある低域通過係数を受け取り該第２結果より減算して第３の結果を発生する第１減算器（該ある低域通過係数は該第１減算器にフィルードバックとして受け取られる）と、該入力データ信号のサンプルの第２のペアを相互に減算して第４の結果を発生する第２減算器と、該第４結果を受け取って該第４結果を第２の因数で除して第５の結果を発生する第１除算器と、該第３結果及び該第５結果を受け取って加算し第６の結果を発生する第２加算器と、該第６結果を受け取り該第６結果を第３の因数で除して第２の低域通過係数を生成する第２除算器とからなり、該第１低域通過係数及び該第２低域通過係数が出力されるウエーブレット変換フィルタが提供される。このウエーブレット変換フィルタにおいて、該乗算器及び／または該除算器の少なくとも１つは例えばシフタからなる。

【００１６】本発明によれば、１対の低域通過係数の減算をして第１の結果を発生する第１減算器と、該第１結果に基づいた第１の入力を、ある高域通過係数から減算して第２の結果を発生する第２減算器と、該第２結果を、ある低域通過係数に基づいた第２入力に加算して第３の結果を発生する第１加算器と、該第２結果を、該ある低域通過係数から減算して第４の結果を発生する第３減算器とからなり、該第３結果に基づいた第１のサンプルと該第４結果に基づいた第２のサンプルを出力するウエーブレット変換フィルタが提供される。このウエーブレット変換フィルタに、該第３結果を受け取り、該第３結果をクリップして該第１サンプルを発生する第１クリップ機構、及び／または、該第１結果を受け取り、該第１結果を第１因数により除して該第１入力を発生する除算器、及び／または、該ある低域通過係数を受け取り、該ある低域通過係数を乗算して該第２入力を生成する乗算器を加えた構成のウエーブレット変換フィルタも本発明により提供される。

【００１７】本発明によれば、複数のデータシンボルを受け取って該複数のデータシンボルをフォーマットされたデータシンボルの集合にフォーマットするフォーマッティングユニットと、該複数のデータシンボルに応じて複数のデシジョンを生成する、該フォーマッティングユニットに接続されたシグニフィカンスユニットと、該シグニフィカンスユニットから該複数のデシジョンを受け取って記憶または出力するメモリ機構とからなる符号化装置が提供される。この符号化装置において、該複数のデータシンボルは例えば複数の係数からなり、該フォーマッティングユニットは、例えば、該複数のデータシンボルを符号－絶対値形式にフォーマットされたデータの集合へ変換するための符号－絶対値ユニットからなり、該符号－絶対値ユニットは、例えば、各データシンボルに対するシグニフィカンス表示を出力するシグニフィカンス判定機構と、該複数のデータシンボルのそれぞれを受け取り、そのデータシンボルに対する符号及び仮数を

生成する符号及び仮数ジェネレータと、該複数のデータシンボルのそれぞれに対するインデックスを生成するインデックスカウンタからなる。該シグニフィカンス判定機構は、例えば、プライオリティエンコーダからなり、あるいは、１対のプライオリティエンコーダと、該１対のプラオリティエンコーダのそれぞれの出力の間で選択することによって該シグニフィカンス表示を出力する選択機構とからなる。

【００１８】また、上記構成の符号化装置に、該複数のフォーマットされたデータシンボルを受け取って記憶する絶対値メモリを追加した構成の符号化装置も本発明により提供される。この符号化装置において、例えば、該複数のフォーマットされたデータシンボルのそれぞれは、シグニフィカンスレベルに基づいて該絶対値メモリに格納される。また、例えば、該絶対値メモリは複数の記憶領域を具備し、該複数の記憶領域のそれぞれは別個のシグニフィカンスレベルのためのデータを格納し、該絶対値メモリは複数のカウンタをさらに具備し、該複数のカウンタのそれぞれは、該フォーマットされたデータシンボルの格納のためのアドレスを提供するため該複数の記憶領域の１つに関係付けられる。

【００１９】

【発明の実施の態様】本発明は、入力データに応じて変換信号を生成する方法及び装置を含む。一実施例において、この変換信号は、可逆ウエーブレット変換によって生成される。本発明はまた、変換信号を、入力データの非損失圧縮されたものを表わすデータへと圧縮するための方法及び装置を含む。一実施例において、本発明は非最小長の可逆フィルタを用いて入力データを分割する。この分割は、多数の１次元フィルタを用いて行なわれるかもしれない。

【００２０】本発明はまた、変換信号の埋め込み符号化（embedded coding）を行なう方法及び装置を含む。本発明の埋め込み符号化は、係数系列の順序付けと変換信号に対するビット・シグニフィカンス埋め込み（bit significance embedding）を含む。

【００２１】本発明はまた、入力データの非損失圧縮データを変換信号へ伸長するための方法及び装置も含む。本発明はまた、非損失性圧縮データの丸めにより入力信号の損失性圧縮も可能である。本発明は、インバース可逆ウエーブレット変換を用いて、変換信号から入力データを生成し、入力データの再現データを得るための方法及び装置も含む。

【００２２】

【実施例】圧縮及び伸長のための方法及び装置について述べる。以下の本発明に関する詳細な説明において、本発明を完全に理解してもらうために、符号化器の種類、ビット数、信号名等々、様々な具体例が示される。しかし、当業者にとって、そのような具体例によらずに本発明を実施し得ることは明白であろう。他方、本発明をい

9

たづらに難解にしないため、周知の構造及びデバイスは詳細にではなくブロック図の形で示される。

【００２３】以下の詳細説明のかなりの部分が、コンピュータメモリ内のデータビットに対する演算のアルゴリズム及び記号表現によって与えられる。これらのアルゴリズム記述及び表現は、データ処理技術分野の当業者によって、その研究の内容を他の当業者に対し最も効率的に伝えるために用いられる手段である。あるアルゴリズムがあり、それが概して、希望する結果に至る自己矛盾のないステップ系列だと考えられるとしよう。これらのステップは、物理量の物理的処理を必要とするものである。必ずという訳ではないが、通常、これらの物理量は記憶、転送、結合、比較、その他処理が可能な電気的または磁気的信号の形をとる。これらの信号をビット、値、要素、記号、文字、用語、数字等で表わすのが、主に慣用上の理由から、時に都合がよいことが分かっている。

【００２４】しかしながら、このような用語は、適切な物理量と関係付けられるべきであって、これら物理量につけた便宜上のラベルに過ぎないということを心に留めるべきである。以下の説明から明らかなように、特に断わらない限り、”処理””演算””計算””判定””表示”等々の用語を用いて論じることは、コンピュータシステムのレジスタ及びメモリ内の物理的（電子的）な量として表現されたデータを処理して、コンピュータシステムのメモリまたはレジスタ、同様の情報記憶装置、情報伝送装置あるいは表示装置の内部の同様に物理量として表現された他のデータへ変換する、コンピュータシステムあるいは同様の電子演算装置の作用及びプロセスを指すものである。

【００２５】本発明はまた、本明細書に述べられる操作を実行する装置に関する。この装置は、必要な目的のために専用に作られてもよいし、あるいは、汎用コンピュータを内蔵プログラムにより選択的に駆動または再構成したものでもよい。本明細書に提示されるアルゴリズム及び表示は、本質的に、いかなる特定のコンピュータやその他装置とも関係がない。様々な汎用マシンを本明細書に教示したところに従ったプログラムと一緒に用いてもよいし、あるいは、必要な方法のステップの実行のためにより特化した装置を作るほうが好都合であるかもしれない。これら多様なマシンに要求される構造は以下の説明より明らかになろう。さらに、本発明は、特定のプログラミング言語と関連付けては説明されない。本明細書において述べるように、本発明の教示するところを実現するために、多様なプログラミング言語を用い得ることが理解されるであろう。

【００２６】＜概要＞本発明は、符号化部及び復号部を持つ圧縮／伸長システムを提供する。符号化部は入力データを符号化して圧縮データを生成する役割を持ち、他方、復号部は前もって符号化されたデータを復号してオ

10

リジナル入力データの再現データを生成する役割を持つ。入力データは、画像（静止画像あるいはビデオ画像）、音声等々の様々なデータ形式のものがあろう。一実施例ではデータはデジタル信号データであるが、デジタル化されたアナログデータ、テキストデータ形式、その他の形式が可能である。データのソースは、符号化部及び／または復号部のためのメモリまたはチャネルであろう。

【００２７】本発明における符号化部及び／または復号部の構成要素は、ハードウエアあるいは、コンピュータシステム上で使われるようなソフトウエアによって実現し得る。本発明は、非損失性圧縮／伸長システムを提供する。本発明はまた、損失性圧縮／伸長を実行するようにも構成し得る。

【００２８】図１は、システムの符号化部の一実施例のブロック図である。なお、システムの復号部はデータフローに沿って逆の順序で動作する。

【００２９】図１において、入力画像データ１０１が可逆ウエーブレット変換ブロック１０２に受け取られる。可逆ウエーブレット変換ブロック１０２の出力はビット・シグニフィカンス埋め込み（bit-significance embedding）ブロック１０３に接続される。可逆ウエーブレット変換ブロック１０２の出力に応答して、ビット・シグニフィカンス埋め込みブロック１０３は少なくとも１つのビットストリームを出力し、このビットストリームはエントロピー符号化器１０４に受け取られる。エントロピー符号化器１０４は、ビット・シグニフィカンス埋め込みブロック１０３からの入力に応じて、符号ストリーム１０７を出力する。

【００３０】一実施例において、ビット・シグニフィカンス埋め込みブロック１０３は、図２に示されるように、符号（sign）ー絶対値フォーマッティングユニット１０９、周波数ベース（frequency-based）コンテキストモデル１０５及び統合空間／周波数（joint space／frequency）コンテキストモデル１０６からなる。一実施例では、統合空間／周波数コンテキストモデル１０６は水平（horizon）コンテキストモデルである。いくつかの実施例では、周波数ベースコンテキストモデル１０５はシグニフィカンス・ツリー（significance tree）モデルである。符号ー絶対値フォーマッティングユニット１０９、周波数ベースコンテキストモデル１０５及び統合空間／周波数（ＪＳＦ）コンテキストモデル１０６は、本発明におけるビット・シグニフィカンス埋め込みを遂行する。符号ー絶対値フォーマッティングユニット１０９の入力は可逆ウエーブレット変換ブロック１０２の出力と接続される。符号ー絶対値フォーマッティングユニット１０９の出力はスイッチ１０８に接続される。スイッチ１０８は、符号ー絶対値フォーマッティングユニット１０９の出力を周波数ベースコンテキストモデル（モデリングブロック）１０５または統合空間／周波数

11

コンテキストモデル（統合空間／周波数モデリングブロック）１０６のいずれかの入力に与えるように接続される。周波数ベースコンテキストモデル（周波数ベース符号化ブロック）１０５及び統合空間／周波数コンテキストモデル（水平順序（horizon order）符号化ブロック）１０６の出力はエントロピー符号化器１０４の入力に接続される。エントロピー符号化器１０４は、出力符号ストリーム１０７を生成する。

【００３１】図１に戻る。本発明においては、後に定義するように、画像データ１０１４は可逆ウエーブレット変換ブロック１０２において受け取られ、可逆ウエーブレットを使って変換符号化され、その画像のマルチ解像度分割を表わす一つの係数系列が生成される。これらの係数はビット・シグニフィカンス埋め込みブロック１０３に受け取られる。

【００３２】ビット・シグニフィカンス埋め込みブロック１０３は、それらの係数を順序付けして符号－絶対値形式へ変換する。そして、このフォーマットされた係数は、それらのシグニフィカンス（significance）（後述）に基づき、様々な埋め込み（embedded）モデリング法の組合せを適用される。本発明においては、フォーマットされた係数は２つの埋め込みモデリング法（例えば周波数ベースモデリングとＪＳＦモデリング）の中のいずれか一方を適用される。

【００３３】一実施例においては、フォーマットされた係数は、周波数ベースモデリングか統合空間／周波数モデリングを適用される。本発明において、入力データが複数のビットプレーン（bitplane）を持つ画像データからなる時には、いくつかのビットプレーンが周波数モデリングによって符号化され、残りのビットプレーンがＪＳＦモデリングにより符号化される。どのビットプレーンにどの方法を使うべきかの決定は、ユーザ・パラメータであってよい。一実施例においては、上位の係数ビットプレーンが順序付けられて本発明の周波数ベースモデリングにより符号化される。本発明の周波数ベースコンテキストモデル法においては、係数ビットのシグニフィカンスの予測はウエーブレットのピラミッド構造と関連している。下位の係数ビットプレーンは順序付けられて本発明の統合空間／周波数コンテキストモデルにより符号化される。このＪＳＦモデリング（例えば水平モデリング）は、周波数ベース符号化に比べ、周波数領域係数関係の相関が小さいビットプレーンに対して有利である。

【００３４】ビット・シグニフィカンス埋め込みの結果が、エントロピー符号化器により符号化すべきデシジョン（decicion）（あるいはシンボル）である。一実施例では、全てのデシジョンが一つの符号化器へ送られる。他の実施例では、デシジョンはシグニフィカンス（significance）によってラベル付けされ、各シグニフィカンス・レベルのデシジョンが別々の複数の（物理または仮想）符号化器によって処理される。

12

【００３５】周波数ベースコンテキストモデル１０５及びＪＳＦモデル１０６より得られたビットストリームは、シグニフィカンス順に、エントロピー符号化器１０４によって符号化される。一実施例では、エントロピー符号化器１０４はバイナリのエントロピー符号化器からなる。ある実施例では、エントロピー符号化器１０４はＱコーダ、米国特許第５，２７２，４７８号に定義されたＢ－コーダ、あるいは、米国特許出願第０８／０１６，０３５号（”Method and Apparatus for ParallelDocoding and Encoding of Data”、１９９３年２月１０日受理）に述べられているような符号化器からなる。Ｑ－コーダについてさらに情報を得るには、Pennebaker, W. B., et al, ”An Overview of the Basic Principles ofthe Q-coder Adaptive Binary Arithmetic, IBM Journal of Research andDevelopment Vol.32, pg. 717-26, 1988 を参照されたい。ある実施例では、単一の符号化器が単一の出力符号ストリームを発生する。他の実施例では、複数の（物理または仮想）符号化器が、複数の（物理または仮想）データストリームを発生する。

【００３６】＜ウエーブレット分割（Wavelet Decomposition）＞本発明は、最初に、可逆ウエーブレットを用いて、画像（画像データとしての）または他のデータ信号の分割を行なう。本発明においては、可逆ウエーブレット変換は、整数係数を持つ信号の非損失性復元が可能な精密復元システムを整数演算で実現する。本発明は、可逆ウエーブレットを用いることにより、限定された精度の演算によって非損失性圧縮を提供できる。画像データに可逆ウエーブレット変換を適用することによって生成される結果は、一つの係数系列である。本発明の一実施例では、可逆ウエーブレット変換はフィルタの集合を用いて実現される。ある実施例では、そのフィルタとは１つの２タップ低域通過フィルタと１つの６タップ高域通過フィルタである。ある実施例では、これらフィルタは加減算（及びハードワイヤのビットシフト）だけで実現される。また、本発明においては、高域通過フィルタは低域通過フィルタの結果を使って出力を生成する。結果として得られる高域通過係数は画素解像度より数ビット分だけ大きく、低域通過係数は画素解像度と同一である。ピラミッド分割では低域通過係数だけが繰り返しフィルタされるため、マルチレベル分割により解像度は増加しない。

【００３７】ウエーブレット変換システムは一対のＦＩＲ分析フィルタ$h_0(n)$，$h_1(n)$と一対のＦＩＲ合成フィルタ$g_0(n)$，$g_1(n)$によって定義される。本発明において、h_0とg_0は低域通過フィルタであり、h_1とg_1は高域通過フィルタである。このウエーブレット変換システムのブロック図が、図３に示されている。

【００３８】図３において、ブロック２０１，２０２により入力信号x(n)に対して分析フィルタh_0，h_1がか

13

けられ、その出力がブロック２０３，２０４で２：１の間引き（臨界的サブサンプリング）を施されることにより、変換信号$y_0(n)$，$y_1(n)$が生成される。この変換信号$y_0(n)$，$y_1(n)$はそれぞれ、本明細書においては低域通過係数、高域通過係数と呼ばれる。これらの分析フィルタとそれらに対応した間引きもしくはサブサンプリングのブロック２０１～２０４は、ウエーブレット変換システムの分析部を構成する。符号化器／復号器２０５，２０６は、変換ドメインにおいて実行される全ての処理論理及びルーチン（例えば、予測、量子化、符号化等々）を含む。

【００３９】図３に示したウエーブレット変換システムは合成部も持ち、この合成部において変換信号はブロック２０７，２０８により１：２のアップサンプリングを施され（各項の後にゼロが挿入される）、ついで合成フィルタ$g_0(N)$，$g_1(n)$（ブロック２０９，２１０）に通される。低域通過係数$y_0(n)$は低域通過合成フィルタg_0（ブロック２０９）に通され、高域通過係数$y_1(n)$は高域通過合成フィルタg_1（ブロック２１０）に通される。フィルタ$g_0(n)$，$g_1(n)$の出力が合成されて

【００４０】

【外１】

$$\hat{x}(n)$$

【００４１】が作られる。

14

【００４２】ある実施例ではダウンサンプリング（サブサンプリング）とアップサンプリングが行なわれるが、他の実施例では、ダウンサンプリング及びアップサンプリングにより不要となる計算が行なわれないようなフィルタが使用される。

【００４３】ウエーブレット変換システムはＺ変換によって記述してもよい。ここにおいて、

【００４４】

【外２】

$$X(Z),\ \hat{X}(Z)$$

【００４５】は入力信号と出力信号であり、Y0（Z），Y1（Z）は低域通過の変換信号と高域通過の変換信号であり、H0（Z），H1（Z）は低域通過分析フィルタと高域通過分析フィルタであり、G0（Z），G1（Z）は低域通過合成フィルタと高域通過合成フィルタである。変換ドメインにおいて修正も量子化もなければ、図３の出力

【００４６】

【外３】

$$\hat{X}(Z)$$

【００４７】は次式により与えられる。

【００４８】

【数１】

$$\hat{X}(Z) = \frac{1}{2}[H_0(Z)G_0(Z)+H_1(Z)G_1(Z)]X(Z)+\frac{1}{2}[H_0(-Z)G_0(Z)+H_1(-Z)G_1(Z)]X(-Z)$$

【００４９】本発明において、上記式における右辺の第２項は”エイリアシング”（折り返し）項と呼ばれるが、これはキャンセルされる。というのは、合成フィルタが分析フィルタの直交ミラーフィルタとされているからである。すなわち、

【００５０】

【数２】

$$\begin{cases} G_0(Z) = H_1(-Z) \\ G_1(Z) = -H_0(-Z) \end{cases}$$

【００５１】フィルタ係数によれば、

【００５２】

【数３】

$$\begin{cases} g_0(n) = (-1)^n h_1(n) \\ g_1(n) = -(-1)^n h_0(n) \end{cases}$$

【００５３】したがって、直交ミラーフィルタのペアに関し、代入すると、出力は次のようになる。

【００５４】

【数４】

$$\hat{X}(z) = \frac{1}{2}[H_0(Z)H_1(-Z)-H_1(Z)H_0(-Z)]X(Z)$$

【００５５】よって、本発明の直交ミラーシステムにおいては、出力は分析フィルタの項だけで決定される。ウエーブレット変換は、フィルタにより生成された出力がフィルタの入力として直接的または間接的に用いられるという点で、変換信号に対し再帰的に適用される。以上説明した実施例においては、低域通過変換成分ｙ０（ｎ）だけが再帰的に変換されるため、当該システムはピラミッド型である。そのようなピラミッドシステムの一例が図９に示されている。

【００５６】Ｚ変換はハードウエア及び／またはソフトウエアのデータ操作を説明するのに便利な表現方法である。Z^{-m}による乗算は、ハードウエアによるｍクロックサイクルの遅延、及び、ソフトウエアによるｍ個前の要素への配列アクセスのモデルである。そのようなハードウエア手段は、メモリ、パイプステージ、シフタ、レジスタ等を含む。

【００５７】本発明において、信号ｘ(n)と信号

15

【００５８】
【外４】

$$\hat{x}(n)$$

【００５９】は、ある乗定数及びある遅延期間までは一致する。すなわち、Ｚ変換では
【００６０】
【数５】

$$\hat{X}(Z) = cZ^{-m}X(Z)$$

【００６１】これは精密復元システムと呼ばれる。しかして、本発明の一実施例において、入力データに最初に適用されたウエーブレット変換は精密に復元可能である。
【００６２】ハダマード変換（Hadamard Transform）を使う本発明の一実施例は精密復元システムであり、その正規化形式はＺ領域で次のように記述される。
【００６３】
【数６】

$$\begin{cases} H_0(Z) = \frac{1}{\sqrt{2}}(1+Z^{-1}) \\ H_1(Z) = \frac{1}{\sqrt{2}}(1-Z^{-1}) \end{cases}$$

【００６４】代入すると、出力は次のとおりである。
【００６５】
【数７】

$$\hat{X}(Z) = Z^{-1}X(Z)$$

【００６６】これが精密復元であることは明らかである。ハダマード変換について更に情報を得るためには、Anil K. Jain, Fundamentals of Image Processing, pg. 155を見られたい。
【００６７】ハダマード変換の可逆型は、本明細書においてＳ－変換と呼ばれる。Ｓ－変換についてさらに情報を得るためには、Said, A. and Pearlman, W.,” ReversibleImage Compression via Multiresolution Representation and PredictiveCoding ”, Dept. of Electrical, Computer and Systems Engineering, Renssealaer Polytechnic Institute, Troy, NY 1993 を参照されたい。ハダマード変換は精密復元変換であるから、下に示す非正規化型（一定の因子だけがハダマード変換と違っている）もまた精密復元変換である。
【００６８】
【数８】

$$\begin{cases} h_0(Z) = \frac{1}{2}(1+Z^{-1}) \\ h_1(Z) = 1-Z^{-1} \end{cases}$$

【００６９】入力信号のサンプルをｘ0，ｘ1とすると、

16

Ｓ－変換はつぎのように当該システムの可逆システムである。
【００７０】
【数９】

$$\begin{cases} y_0(0) = \lfloor (x(0)+x(1))/2 \rfloor \\ y_1(0) = x(0)-x(1) \end{cases}$$

【００７１】上記式中の記号
【００７２】
【外５】

$$\lfloor \cdot \rfloor$$

【００７３】は、切り捨てる、あるいは丸めることを意味し、時にフロアー関数（floorfunction）と呼ばれる。同様に、シーリング関数（ceiling function）
【００７４】
【外６】

$$\lceil \cdot \rceil$$

【００７５】は最も近い整数へ切り上げることを意味する。
【００７６】このシステムが可逆であることの証明は、近似により失われる情報がｘ(0)＋ｘ(1)の最下位ビットだけであるという事実から得られる。しかしながら、ｘ(0)＋ｘ(1)とｘ(0)－ｘ(1)の最下位ビットは同一であるので、これは高域通過出力$y_1(0)$から再生することができる。換言すれば、次式のとおりである。
【００７７】
【数１０】

$$\begin{cases} x(0) = y_0(0) + \lfloor (y_1(0)+1)/2 \rfloor \\ x(1) = y_0(0) - \lfloor (y_1(0)-1)/2 \rfloor \end{cases}$$

【００７８】Ｓ－変換は、最小長（minimal length）の可逆フィルタを用いる非オーバーラップ（non-overlapping）変換である。最小長フィルタは、２タップのフィルタのペアからなる。最小長変換は良好なエネルギー集中を得られない。最小長フィルタは、その長さがフィルタの数に等しいので、非オーバーラップ変換を実現する。オーバーラップ変換は、フィルタ数より大きな長さのフィルタを少なくとも１つ用いる。長い（非最小長の）フィルタを使うオーバーラップ変換は、より良好なエネルギー集中を得ることができる。本発明は、オーバーラップ変換を可能にする非最小長の可逆フィルタを提供する。
【００７９】精密復元システムのもう一つの例は、次に示すＺ領域定義を持つTwo/Six（ＴＳ）－変換からなる。
【００８０】
【数１１】

17

$$\begin{cases} H_0(Z)=\frac{1}{\sqrt{2}}(1+Z^{-1}) \\ H_1(Z)=\frac{1}{8\sqrt{2}}(-1-Z^{-1}+8Z^{-2}-8Z^{-3}+Z^{-4}+Z^{-5}) \end{cases}$$

【００８１】代入すると、出力は次の通りであり、これは精密復元変換である。

【００８２】

【数１２】

$$\hat{X}(Z)=2Z^{-3}X(Z)$$

【００８３】ＴＳ－変換の有理非正規化型は次のとおりである。

18

【００８４】

【数１３】

$$\begin{cases} h_0(Z) = \frac{1}{2}(1+Z^{-1}) \\ h_1(Z) = \frac{1}{8}(-1-Z^{-1}+8Z^{-2}-8Z^{-3}+Z^{-4}+Z^{-5}) \end{cases}$$

【００８５】$\dot{x}(0)$，$x(1)$，...，$x(5)$が信号の６サンプルであるとき、最初の３つの低域通過係数$y_0(0)$，$y_0(1)$，$y_0(2)$と最初の高域通過係数$y_1(0)$は次式で示される。

【００８６】

【数１４】

$$\begin{cases} y_0(0) = \lfloor (x(0)+x(1))/2 \rfloor \\ y_0(1) = \lfloor (x(2)+x(3))/2 \rfloor \\ y_0(2) = \lfloor (x(4)+x(5))/2 \rfloor \end{cases}$$

$$y_1(0)=\left\lfloor \left(-(x(0)+x(1))\right)+8(x(2)-x(3))+(x(4)+x(5))/8 \right\rfloor$$

【００８７】しかし、ＴＳ－変換の有理非正規化型をそのまま具体化したのでは可逆でない。次に述べる例で、そのような構成が局所的に非可逆であることを明らかにする。グローバルケースのための例としては、もっと長い系列を作ることができる。$y_0(0)$と$y_0(2)$を計算するために丸めを行なうため、

$-(x(0)+x(1)+)+(x(4)+x(5)) \neq -y_0(0)+y_0(2)$

であるから、ローカル情報を使うとき当該変換は可逆でない。

【００８８】例えば、$x(0)=1$，$x(1)=1$，$x(2)=3$，$x(3)=1$，$x(4)=1$，$x(5)=1$であるときには、

【００８９】

【数１５】

$$\begin{aligned} y_0(0)&=\lfloor (1+1)/2 \rfloor = 1 \\ y_0(1)&=\lfloor (3+1)/2 \rfloor = 2 \\ y_0(2)&=\lfloor (1+1)/2 \rfloor = 1 \\ y_1(0)&=\lfloor \lfloor -(1+1)+8(3-1)+(1+1) \rfloor /8 \rfloor = \lfloor (-2+16+2)/8 \rfloor = 2 \end{aligned}$$

【００９０】また、$x(0)=1$，$x(1)=2$，$x(2)=4$，$x(3)=1$，$x(4)=1$，$x(5)=1$であるときには、

【００９１】

【数１６】

$$\begin{aligned} y_0(0)&=\lfloor (1+2)/2 \rfloor = 1 \\ y_0(1)&=\lfloor (4+1)/2 \rfloor = 2 \\ y_0(2)&=\lfloor (1+1)/2 \rfloor = 1 \\ y_1(0)&=\lfloor \lfloor -(1+2)+8(4-1)+(1+1) \rfloor \rfloor /8 = \lfloor (-3+24+2)/8 \rfloor = \lfloor 23/8 \rfloor = 2 \end{aligned}$$

【００９２】$y_0(0)$，$y_0(1)$，$y_1(0)$は入力$x(0)$...$x(5)$の異なった２つの組に対して同一であるから、この変換は可逆でない。$y_0(0)$，...，$y_1(0)$を与えられたとき、このローカル情報から、その２つの組のいずれが入力されたのか判定できないからである。（全係数からのグローバル情報を用いるときに、当該変換が可逆でないとは証明できないことに注意されたい。）

さて、異なった高域通過フィルタ作用を提供する可逆ＴＳ－変換を考えよう。この変換を、ここではＲＴＳ－変換と呼ぶ。

【００９３】$x(0)$，$x(1)$，$x(2)$，$x(3)$，$x(4)$，$x(5)$が信号の６サンプルであるとき、初めの３つの低域通過係数$y_0(0)$，$y_0(1)$，$y_0(2)$と最初の高域通過係数$y_1(0)$は次式で与えられる。

【００９４】

【数１７】

$$\begin{cases} y_0(0) = \lfloor (x(0)+x(1))/2 \rfloor \\ y_0(1) = \lfloor (x(2)+x(3))/2 \rfloor \\ y_0(2) = \lfloor (x(4)+x(5))/2 \rfloor \end{cases}$$

$$y_1(0) = \lfloor (-\lfloor (x(0)+x(1))/2 \rfloor + 4(x(2)-x(3)) + \lfloor (x(4)+x(5))/2 \rfloor)/4 \rfloor$$
$$= \lfloor (-y_0(0)+4(x(2)-x(3))+y_0(2))/4 \rfloor$$

【００９５】

【数１８】

$$x(2)-x(3) = y_1(0) - \lfloor -(y_0(0)+y_0(2))/4 \rfloor$$

【００９６】上式が成り立つので、ｘ(2)－ｘ(3)は完全に既知となった。前記（数１７）のｙ0(1)と、ｘ(2)－ｘ(3)とｘ(2)－ｘ(3)とが上のように決まれば、ｘ(0)＋ｘ(1)とｘ(0)－ｘ(1)の最下位ビットは同一であるから、ｘ(2)とｘ(3)は復元できる。すなわち、次式のとおりである。

【００９７】

【数１９】

$$d(0) = x(2)-x(3) = y_1(0) - \lfloor (-y_0(0)+y_0(2)/4) \rfloor$$
$$x(2) = y_0(1) + \lfloor (d(0)+1)/2 \rfloor$$
$$x(3) = y_0(1) + \lceil (d(0)-1)/2 \rceil$$

【００９８】プログラミング言語”Ｃ”で実現した、ＲＴＳ－変換のためのフィルタの例を図５７乃至図６０に示す。図５７及び図５８に示すものは１レベル分割ＲＴＳ変換のためのフォワードフィルタ及びインバースフィルタであり、図５９及び図６０に示すものは２レベル分割ＲＴＳ変換のためのフォワードフィルタ及びインバースフィルタである。

【００９９】なお、数学的には

$1/8\ (-1-Z^{-1}+8Z^{-2}-8Z^{-3}+Z^{-4}+Z^{-5})$

と、

$1/4\ (1/2\ (-1-Z^{-1})+4\ (Z^{-2}-Z^{-3})+1/2\ (Z^{-4}+Z^{-5}))$

は、無限精度演算によって行なう時には同一である。２番目の式が可逆フィルタを表わしている理由は、整数演算で実際に実施してみれば明白である。低域通過フィルタと高域通過フィルタをハードウエアにより実現した典型例を、図２９及び図３０に関連して説明する。

【０１００】なお、Ｓ－変換及びＲＴＳ－変換の両方において、低域通過フィルタは入力信号ｘ(n)のレンジが出力信号ｙ0(n)のレンジと同一になるように構成される。例えば、信号が８ビットの画像であるときには、低域通過フィルタの出力も８ビットである。このことは、低域通過フィルタが連続的に適用されるピラミッドシステムにとって重要な特性である。というのは、従来システムでは、出力信号のレンジが入力信号のレンジより大きく、そのことがフィルタの連続的適用を難しくしていたからである。さらに、低域通過フィルタは２つのタップしか持たないため、このフィルタは非オーバーラップフィルタになる。この特性は、後述のように、フィルタをハードウエアで実現する上で重要である。

【０１０１】ＲＴＳ－変換に関し、一実施例では低域通過フィルタ及び高域通過フィルタは次のように定義される。

【０１０２】

【数２０】

$$\begin{cases} h_0(Z) = \frac{1}{2}(1+Z^{-1}) \\ h_1(Z) = \frac{1}{4}\left(-\frac{1}{2}\left(1+Z^{-1}\right)+4\left(Z^{-2}-Z^{-3}\right)+\frac{1}{2}\left(Z^{-4}+Z^{-5}\right)\right) \end{cases}$$

【０１０３】このように、低域通過フィルタの出力は高域通過フィルタにおいて２回（第１項と第３項で）使われる。したがって、高域通過フィルタの結果に到達するために、ほかに２つの加算しか行なう必要がない。

【０１０４】多くのオーバーラップ非最小長可逆フィルタが本発明で用いられてもよい。非オーバーラップ最小長可逆フィルタでフィルタリングを行なう変換システムのフォワード変換とインバース変換の説明を図４に示す。例えば、次に示す種類のフィルタが本発明に用いられてもよい。整数Ｌ≧ｚに対し、

【０１０５】

【数２１】

21　　22

$$d(0)=x\left(2\left(\lfloor L/2\rfloor+1\right)\right)-x\left(2\left(\lfloor L/2\rfloor+1\right)+1\right)$$

$$y_0(0)=\lfloor (x(0)+x(1))/2\rfloor$$

$$y_0(1)=\lfloor (x(2)+x(3))/2\rfloor$$

$$y_0(L-1)=\lfloor (x(2\lfloor (L-1)/2\rfloor)+x(2\lfloor (L-1)/2\rfloor+1))/2\rfloor$$

$$y_1(0)=\frac{\sum_{i=0}^{\lfloor L/2\rfloor}a_i y_0(i)+bd(0)+\sum_{j=\lfloor L/2\rfloor+2}^{L-1}c_i y_0(j)}{k}$$

【０１０６】この高域通過フィルタの長さは２Ｌである。Ｌが奇数ならば、フィルタはより対称フィルタに近くなろう。a_i，ｂ，c_i，ｋが整数で、ｋ≦ｂのときは、フィルタは可逆である。a_i，ｂ，c_i，ｋが２のべき乗（あるいは、ある２のべき乗の負値もしくは補数）であれば、フィルタの構成が簡単になる。（a_i，c_iの値と関係なく）ｋ＝ｂならば、高域通過フィルタ出力y_1のレンジは最小になる。各a_iについて、a_i＝－c_iとなるc_iが丁度１つ存在するときには、高域通過フィルタは一定した入力に対し全く応答しない。ｊ－(L-1)＝ｉの時にａｉ＝－ｃｉならば、フィルタは、より対称フィルタに近くなろう。

【０１０７】もう一つの有用な特性が次式で表わされる。

【０１０８】

【数２２】

$$\sum_{i=0}^{\lfloor L/2\rfloor}\left[(a_i)(2i)^m+(a_i)(2i+1)^m\right]+(b)\left(2\left(\lfloor L/2\rfloor+1\right)\right)^m$$

$$-(b)\left(2\left(\lfloor L/2\rfloor+1\right)+1\right)^m+\sum_{i=\lfloor L/2\rfloor+2}^{L-1}\left[c_j(2_j)^m+c_j(2_j+1)^m\right]=0$$

【０１０９】この特性によって、高域通過フィルタが、ｍ＝１の時に直線的に変化する入力に対し応答しなくなり、ｍ＝２の時に二次曲線的に変化する入力信号に対し応答しなくなる、等々となる。ここでｍはモーメント条件である。この特性が、ＲＴＳ－変換のエネルギー集中がＳ－変換より優れていることの主な根拠である。

【０１１０】フィルタは可逆性のための最低条件を充足しなければならないけれども、用途が違えば、他の特性のいずれも満たさないフィルタ、いくつかを満たすフィルタ、あるいは全部を満たすフィルタを用いてよい。いくつかの実施例においては、次に例示する高域通過フィルタの中の１つが用いられる。本発明を難解にしないため、これらのフィルタは、フィルタの有理型の整数係数を単に表にして表わす形式で下に示す。

【０１１１】

1	1	-4	16	-16	4	4	-1	-1	
1	1	-3	-3	8	-8	3	3	-1	-1
-1	-1	0	0	16	-16	0	0	1	1
-1	-1	4	4	-16	-16	256	-256	16	16	-4	-4	1	1
3	3	-22	-22	128	-128	22	22	-3	-3

最後のフィルタは（Two／Ten）ＴＴ－フィルタであり、３次増加関数に対して応答しないという特性を有する。なお、22=16+2x3、3=2+1であるので、このフィルタは合計７つの加減算によって実現できることに注意されたい。

【０１１２】フィルタの厳密な可逆性要件は、次のことに留意することにより緩和できる。高域通過係数は、ある順序で符号化され復号される。前に復号された高域通過係数に対応する画素値が正確に知れれば、それを現在の高域通過フィルタリングで使うことができる。例えば、下記のフィルタは、ラスター順が用いられる時に使用できる。

【０１１３】

【数２３】

$$H_1(Z)=\left\lfloor\frac{1}{4}\left(\left\lfloor-\frac{1}{2}\left(1+Z^{-1}\right)\right\rfloor+\left\lfloor\frac{1}{2}\left(8\left(Z^{-2}-Z^{-3}\right)+\left(Z^{-4}+Z^{-5}\right)\right)\right\rfloor\right)\right\rfloor$$

【０１１４】単一の固定高域フィルタを使用する必要はない。適応型フィルタを使ってもよいし、あるいは複数のフィルタを使うこともできる。複数のフィルタの適応化ないし選択のために使われるデータは、ある特定の逆フィルタリング動作に先立って復号器内で入手可能なデータに限定されなければならない。

23

【０１１５】複数のフィルタを使用する一方法は、高域通過係数をプログレッシブ処理することである。１つおきの高域通過フィルタリング演算（$y_1(0)$，$y_1(2)$，$y_1(4)$，...）は、ＲＴＳ高域通過フィルタのような可逆フィルタを用いて最初に処理してよい。残りのフィルタリング操作（$y_1(1)$，$y_1(3)$，$y_1(5)$，...）には、最高６タップの非可逆フィルタを用いてよい。というのは、フィルタのオーバーラップ部分への入力の正確な値が知れているからである。例えば、次に示すフィルタのどれでも使用してよい。

【０１１６】

-1　3　-3　1

-1　4　-4　1

-3　8　-8　3

1 -5　10 -10　5　-1

1 -4　8　-8　4　-1

いくつかの実施例では、高域通過フィルタは予測／補間（内挿）操作に置き換えられるかもしれない。予測器／補間器は、特定の予測／補間操作の前に復号器において入手可能なデータを利用し、１対の入力間の差分を予測する。予測した差分は入力の実際の差分から差し引かれ、その差が出力される。一実施例では、ＤＰＣＭ、プログレッシブ符号化あるいは空間領域符号化に使われている従来の予測方法が使用される。

【０１１７】本発明の低域通過フィルタ及び高域通過フィルタを用いて、マルチ解像度分割が行なわれる。分割レベル数は可変であり任意数でよいが、しかしながら、現在のところ分割数は２乃至５レベルである。

【０１１８】例えば、可逆ウエーブレット変換が再帰的に１つの画像に適用されるときは、第１レベルの分割は最も細かいディテールもしくは解像度に対し作用する。第１分割レベルでは、画像は４つのサブ画像（すなわちサブバンド）に分割される。各サブバンドは、１つの空間周波数帯域を表わしている。第１レベルのサブバンドはＬＬ0，ＬＨ0，ＨＬ0，ＨＨ0と称される。原画像を分割するプロセスは、水平，垂直両次元における２：１のサブサンプリングを含むので、図５に示されるように、第１レベルのサブバンドＬＬ0，ＬＨ0，ＨＬ0，ＨＨ0はそれぞれ入力が持っている画像の画素数（または係数）の４分の１の数の係数を持っている。

【０１１９】サブバンドＬＬ0は、水平方向の低い周波数の情報と垂直方向の低い周波数の情報を同時に含んでいる。一般に、画像エネルギーの大部分が当該サブバンドに集中している。サブバンドＬＨ0は、水平方向の低い周波数の情報と垂直方向の高い周波数の情報（例えば水平方向エッジ情報）を含んでいる。サブバンドＨＬ0は、水平方向の高い周波数の情報と垂直方向の低い周波数の情報（例えば垂直方向エッジ情報）を含んでいる。サブバンドＨＨ0は、水平方向の高い周波数の情報と垂直方向の高い周波数の情報（例えばテクスチャもしくは

24

斜めエッジの情報）を含んでいる。

【０１２０】この後に続く第２、第３、第４の分割レベルは、前レベルの低周波ＬＬサブバンドを分割することによって作られる。第１レベルの当該サブバンドＬＬ0が分割されることによって、そこそこ精細な第２レベルのサブバンドＬＬ1，ＬＨ1，ＨＬ1，ＨＨ1が作られる（図６）。同様に、サブバンドＬＬ1が分割されることによって、粗い第３レベルのサブバンドＬＬ2，ＬＨ2，ＨＬ2，ＨＨ2が生成される（図７）。また、図８に示されるように、サブバンドＬＬ2が分割されることにより、より粗い第４レベルのサブバンドＬＬ3，ＬＨ3，ＨＬ3，ＨＨ3が作られる。２：１のサブサンプリングにより、第２レベルの各サブバンドは、原画像の１６分の１の大きさである。このレベルの各サンプル（つまり画素）は、原画像中の同一位置のそこそこ細いディテールを表現する。同様に、第３レベルの各サブバンドは、原画像の３４分の１の大きさである。第３レベルでの各画素は、原画像中の同一位置のかなり粗いディテールを表現する。また、第４レベルの各サブバンドは、原画像の２５６分の１の大きさである。

【０１２１】分割された画像はサブサンプリングによる物理的に原画像より小さいので、原画像の格納のために使用されるメモリを利用して、分割サブバンド全部を格納できる。換言すれば、３レベル分割の場合、原画像と分割サブバンドＬＬ0，ＬＬ1は捨てられ、保存されない。

【０１２２】粗いディテールのサブバンド成分と、その次に精細なディテールのサブバンド成分との間には親－子の関係が存在する。

【０１２３】４つのサブバンド分割レベルだけを示したが、個別のシステムの条件に応じて、それ以上のレベルを生成することも可能である。また、ＤＣＴのような他の変換あるいは一次元配置のサブバンドによって、異なった親子関係が定義されてもよい。

【０１２４】マルチ解像度分割のプロセスは、図９すようなフィルタ系列を使って行ない得る。長さＬの一次元信号を表わす入力信号は、フィルタユニット４０１，４０２による低域通過フィルタリング及び高域通過フィルタリングの後、ユニット４０３，４０４によって２：１にサブサンプリングされる。ユニット４０３から出力されるサブサンプル信号は、ユニット４０５，４０６による低域通過フィルタリング及び高域通過フィルタリングの後、ユニット４０７，４０８により２：１にサブサンプリングされる。サブバンド成分Ｌ，Ｈがユニット４０７，４０８の各出力に現われる。同様に、ユニット４０７の出力信号は、ユニット４０９，４１０により低域通過フィルタリング及び高域通過フィルタリングを施された後、ユニット４１１，４１２によりそれぞれサブサンプリングされる。サブバンド成分Ｌ，Ｈがユニット４１１，４１２の各出力に現われる。上に述べたように、本

発明の一実施例においてサブバンド分割に用いられるフィルタは、水平周波数帯域と垂直周波数帯域を低周波帯域と高周波帯域へ分割するためのデジタル直交ミラーフィルタである。

【０１２５】図１０は、二次元の２レベル変換を示している。図１１も、図２９及び図３０に示したような一次元フィルタを使って実現した二次元２レベル変換を示している。一次元フィルタ（４５１～４５６）は、サブサンプリングにより不要ににる演算を避けるため１画素位置おきに適用される。一実施例では、一次元フィルタは演算を低域通過フィルタ演算と高域通過フィルタ演算との間に割り振る。

【０１２６】したがって、本発明は、非最小長のオーバーラップ可逆フィルタが使用される圧縮・伸長システムを提供する。図１２は、そのようなシステムの一実施例のブロック図である。図１２において、最初に階層的分割が行なわれる（ブロック４６１）。この階層的分割の結果は圧縮器４６２へ送られて圧縮される。ここで実行される圧縮には、ベクトル量子化、スカラー量子化、ゼロ・ランレングス計数、ハフマン符号化等々が含まれよう。圧縮器４６２の出力は、オリジナル入力データの圧縮されたものを表わすデータからなる。伸長器４６３は、そのデータを、いつか受け取って伸長することになろう。本発明は、次に、非最小長オーバーラップ可逆フィルタを用い逆分割を行なって（ブロック４６４）、オリジナルデータの復元データを生成する。

【０１２７】本発明の可逆ウエーブレットフィルタは、図１３に示すように、典型的な分析及び強調システムにも使用し得る。図１３において、非最小長オーバーラップ可逆ウエーブレットフィルタを使って、入力データに対し階層的分割が行なわれる（ブロック４７１，４７３）。分析ユニット４７２は、フィルタにより生成された係数を受け取り、それら係数をデシジョンに分類する。すなわち、係数を完全に符号化するのではなく、重要な情報だけが抽出される。例えば、文書保管システムにおいては、空白のページが最も粗い低域通過サブバンドだけを使って認識されよう。もう一つ例を挙げれば、ある特定のサブバンドの高域通過情報だけを用いて、テキスト画像と自然シーン画像とを区別する。階層的分割は、最初に粗いサブバンドによって粗いレジストレーション（registration）が行なわれるような、多数の画像のレジストレーションのためにも使用し得る。もう一つの実施例においては、係数は強調処理またはフィルタリングを施され（ブロック４７４）、次に逆分割が行なわれる（ブロック４７５）。鮮明化、エッジ強調、ノイズ制御等々を階層的分割を使って行なうこともできる。このように、本発明は、統合時間／空間領域及び周波数領域の分析並びにフィルタリング／強調処理システム用のウエーブレット変換を提供する。

【０１２８】＜ビット・シグニフィカンス埋め埋み（Bit-Significance Embedded ）符号化＞本発明では、ウエーブレット分割の結果として生成される係数がエントロピー符号化される。本発明においては、係数は最初に埋め込み符号化（embeddedcoding）を施されるが、この符号化では、視覚的に重要な順に係数が順序付けられ、また、より一般的には、何等かの誤差規準（例えば、歪み規準）を考慮して係数が順序付けられる。誤差または歪みの規準には、ピーク誤差と平均２乗誤差（ＭＳＥ）が含まれる。また、順序付けは、ビット・シグニフィカンス空間配置（bit-significance spatial location）、データベース照会のための妥当性及び方向（垂直、水平、斜め等）を優先させるように行なうことができる。本発明は、複数の埋め込み符号化法を利用し、あるシグニフィカンス・レベルの一部の係数はある符号化法で符号化されるが、残りの係数は別の符号化法で符号化される。本発明においては、周波数ベースモデリングと統合空間／周波数モデリングとが、それぞれ本発明のウエーブレット変換により生成された係数の符号化に利用される異なった２つの埋め込み符号化システムである。周波数ベースモデリングは、低い周波数の１係数を符号化する時に高い周波数の多数の係数を予測する必要がある。統合空間／周波数モデリングは、既知の周波数帯域と近傍画素（またはデータ）を両方利用する。統合空間／周波数モデリングの一例は、本明細書において水平（horizon）モデリングと呼ばれる。

【０１２９】データはまず符号ー絶対値形式にフォーマットされ、次に該データはシグニフィカンス（singificance）に基づいて格納される。特定のシグニフィカンス規準を考慮してデータが格納された後、該データは符号化される。周波数ベース符号化及び水平符号化は両方ともビット・シグニフィカンス順序付けをベースにしてよいが、異なるイベント符号化方法を用いる。

【０１３０】ある信号がｘ(n)毎にＲビットの精度で表現されているとした場合、本発明の埋め込み符号化は、その信号の各ｘ(n)の最上位ビット（または複数ビット）を符号化し、次にその下位のビット（または複数ビット）を符号化し、さらに下位のビットというように符号化する。例えば、視覚的に決まる順序付けの場合、中央部分でコーナー沿いまたは縁近傍より高い品質を要求する画像（例えば、ある種の医用画像）は、中央部分の画素の下位ビットが周辺部の画素の上位ビットより先に符号化されるような符号化がなされよう。

【０１３１】ビット・シグニフィカンス歪み（bit significance distortion）規準に基づいた埋め込みシステムの場合、データのバイナリ値は絶対値によって順序付けられる。値が負でない整数の場合（例えば、画素の輝度に関してそうなる）、採用し得る順序はビットプレーン順（例えば、最上位ビットプレーンから最下位ビットプレーンへの順番）である。２の補数の負整数も許容される実施例では、符号ビットの埋め込み順序は、整数の

絶対値の最初の非ゼロのビットと同じである。したがって、1つの非ゼロビットが符号化されるまで、符号ビットは考慮されない。その結果、本発明のビット・シグニフィカンス埋め込みシステムにおいて1つのイベントの可能値は、符号ビットが符号化される前では3元である。この三元イベントは、”非有意（not significant）”、”積極的有意（positivesignificant”、”消極的有意（negative significant）”である。例えば、符号ー絶対値表記法を使うと、－7の16ビット数は

1000000000000111

である。ビットプレーンベースで、初めの12デシジョンは”非有意”となろう。最初の”1”ービットは13番目のデシジョンで生じる。13番目のデシジョンは”消極的有意”となろう。符号ビットが符号化された後、その可能なイベントは2つ、すなわち0，1に減る。14番目と15番目のデシジョンは共に”1”である。

【0132】本発明の一実施例においては、係数を記録するためにリストが用いられる。一実施例では、各係数に関係付けられた1ビットのフラグ（本明細書ではグループフラグと呼ぶ）で、符号ビットが符号化前の係数と、符号ビットが符号化済みの係数を区別する。もう一つの実施例では、フラグビットの代わりに2つ以上のリストを使用できる。別の実施例では単一のリストがフラグ無しで用いられる。

【0133】もう一つの実施例では、リストは使用されない。1つの係数に対する全てのデシジョンが生成され、シグニフィカンスによってラベル付けされた後に、次の係数に対するデシジョンが生成される。こうすることで、リストに係数全部は格納しなくともよくなる。

【0134】＜符号化プロセス及び復号プロセス＞次に説明する図44乃至図55のフローチャートは、本発明の符号化プロセス及び復号プロセスを表わしている。図44及び図45は、本発明の符号化器の変換及びモデリングプロセスを表わすフローチャートである。図44において、符号化器の変換及びモデリングプロセスは、初めに入力データを獲得する（処理ブロック2501）。入力データ獲得後、本発明は可逆ウエーブレットフィルタをかける（処理ブロック2502）。

【0135】次に、別のレベルの分割を要するか判定する（処理ブロック2503）。別のレベルの分割を要するときには、処理ブロック2504へ進み、そこで直前の分割により得られたＬＬ係数に可逆フィルタがかけられ、そして処理ブロック2503へ戻る。このようにして、本発明は任意のレベル数の分割が行なえるようにする。

【0136】別のレベルの分割を要しないときには、処理ブロック2506へ進み、各係数のためのグループフラグをＡーグループに初期化する。グループフラグを初期化した後、Ａーパス用のビットプレーンS_Aは最上位ビットプレーン（ｍａｘ）とされる（処理ブロック2507）。次にＢーパス用のビットプレーンS_Bはその1つ下位のビットプレーン（ｍａｘ－1）とされる（処理ブロック2508）。次に、Ａーパス用のビットプレーンS_Aを周波数ベースモデルを用いて符号化すべきか否かを判定する（処理ブロック2509）。ビットプレーンS_Aを周波数ベースモデルで符号化すべきであれば、処理ブロック2510へ進み、各係数は周波数ベースモデルでモデル化されてからエントロピー符号化される。他方、ビットプレーン S_A が周波数ベースモデルで符号化されるべきでなければ、処理ブロック2511に進み、各係数は統合空間／周波数モデルでモデル化されてからエントロピー符号化される。

【0137】いずれの場合でも、その後に処理ブロック2512へ進み、ビットプレーンS_Aがゼロ以上であるか判定することにより、当該ビットプレーンが最後のビットプレーンであるか否かを表示する。ビットプレーンS_Aがゼロ以上ならば、処理ブロック2509へループバックする。他方、ビットプレーンS_Aがゼロ以上でないときには、処理ブロック2513へ進み、ビットプレーンS_Bがゼロ以上であるか判定することにより、該ビットプレーンがＢーパスを受けるべき最後のビットプレーンであるか判定する。ビットプレーンS_Bがゼロ以上ならば、処理ブロック2509に進む。しかし、ビットプレーンS_Bがゼロ以上でなければ、処理ブロック2514に進み、符号化データがチャネルへ転送されるかメモリに格納される。符号化データを格納または転送した後、本発明の符号化器の変換及びモデリングプロセスは終了する。

【0138】図46及び図47は、本発明の復号器の変換及びモデリングプロセスを示す。図46において、本発明の復号器の変換及びモデリングプロセスは、まず符号化データの取り込みを行なう（処理ブロック2601）。この符号化データはチャネルまたはメモリ、あるいは他の伝送システムより受け取られるであろう。符号化データを取り込んだ後に、各係数のグループフラグがＡーグループに初期化される（処理ブロック2602）。この初期化に続いて、Ａーパス用ビットプレーンS_Aは最上位ビットプレーン（ｍａｘ）とされ（処理ブロック2603）、次にＢーパス用ビットプレーンS_Bはその1つ下位のビットプレーン（ｍａｘ－1）とされる（処理ブロック2604）。そして、各係数の値が初期値のゼロに設定される（処理ブロック2605）。

【0139】各係数の値をゼロに初期化した後、ビットプレーンS_Aが周波数ベースモデルを用いて復号されるべきか否かを判定する（処理ブロック2606）。ビットプレーンS_Aを周波数ベースモデルで復号すべきであれば、処理ブロック2607へ進み、各係数は周波数ベースモデルでモデル化されたのちエントロピー復号される。ビットプレーンS_Aを周波数ベースモデルで復号すべきでないときには、処理ブロック2608へ進み、各

係数は統合空間／周波数モデルでモデル化されてからエントロピー復号される。

【０１４０】各係数がモデル化された後、処理は処理ブロック２６０９へ進み、ビットプレーンS_Aが最後のビットプレーンであるか判定するために、ビットプレーンS_Aがゼロ以上であるか調べる。ビットプレーンS_Aがゼロ以上ならば、処理ブロック２６０６に進む。他方、ビットプレーンビットプレーンS_Aがゼロ以上でなければ、Ｂ－パスのビットプレーンS_Bがゼロ以上であるか判定することにより、ビットプレーンS_BがＢ－パス用の最後のビットプレーンであるか否かを表示する（処理ブロック２６１０）。最後のビットプレーンであるならば、次の復号のために処理ブロック２６０６へ進む。他方、Ｂ－パス用ビットプレーンS_Bがゼロ以上でないならば、最も粗い分割レベルによる係数に逆（inverse）可逆フィルタがかけられる（処理ブロック２６１１）。次に、全てのレベルの逆フィルタ処理が済んだか判定する（処理ブロック２６１２）。済んでいなければ、残っている最も粗い分割レベルに関する係数に、逆可逆フィルタがかけられる（処理ブロック２６１３）。そして処理ブロック２６１２に戻って、全てのレベルの逆フィルタ処理が済んだか再度調べる。

【０１４１】全レベルが逆フィルタ処理済みとなったならば、処理ブロック２６１４に進み、復元データの格納または伝送が行なわれる。

【０１４２】図４８及び図４９は、各係数のモデリングプロセスの一実施例を示す。図示のプロセスは、周波数ベースモデリングまたはＪＳＦモデリングのモデリングプロセスと符号化または復号を表わしている。すなわち、４つの処理ブロック（２５０７，２５０８，２６０７，２６０８）のそれぞれは、図４８及び図４９のモデリングプロセスを用いて実現し得る。図４８において、最初のプロセスは、モデリングが１パスで行なわれるべきか否かの判定（処理ブロック２７０１）で始まる。モデリングが１パスでなされるべきでないときは、ビットプレーンS_AがビットプレーンS_Bより大きいか判定する（処理ブロック２７０２）。大きくなければ、プロセスは処理ブロック２７０３へ移り、Ａ－パスが実行されるべきでないことを表示するためフラグ（do_A_flag）がクリアされる。ビットプレーンS_AがビットプレーンS_Bより大きいときは、処理ブロック２７０４に進み、フラグ（do_A_flag）はセットされ、Ａ－パスが実行されることを表示する。

【０１４３】処理ブロック２７０３または２７０４の次に、処理ブロック２７０５に進み、ビットプレーンS_BがビットプレーンS_Aと等しいか判定する。等しくなければ、本発明はＢ－パスを発生させないためにフラグ（do_B_flag）をクリアし（処理ブロック２７０５ｂ）、そして処理ブロック２７０７に進む。ビットプレーンS_BがビットプレーンS_Aと等しいときには、フラグdo_B_flagがセットされてＢ－パスが実行されることを表示し（処理ブロック２７０６）、処理はまた処理ブロック２７０７に進む。

【０１４４】処理ブロック２７０７で、Ａ－パスフラグがセットされており、ゼロツリー（zerotree）モデリングが行なわれるべきか判定する。フラグが、Ａ－パスが発生しゼロツリーモデリングが行なわれることを表示しているときには、”確定／未確定”（determined／undetermined）フラグが、各係数に対し”未確定”状態に初期化され（処理ブロック２７０８）、処理ブロック２７０９へ進む。他方、Ａ－パス表示フラグもゼロツリーモデリング表示もセットされていないときには、処理ブロック２７０９へ直接進む。処理ブロック２７０９において、最初の係数が変数Ｃに設定される。

【０１４５】最初の係数が変数Ｃに割り当てられると、Ｂ－パス表示フラグがセットされているか判定する（処理ブロック２７１９）。Ｂ－パス表示フラグ（do_B_flag）がセットされているならば、本発明は係数Ｃに対しＢ－パスを実行し（処理ブロック２７１０）、処理ブロック２７１１へ進む。他方、Ｂ－パスフラグがセットされていないときには、Ｃに対してＢ－パスは実行されず、処理は処理ブロック２７１１へ直接進む。

【０１４６】次に、Ａ－パス表示フラグ（do_A_flag）がセットされているか判定する（処理ブロック２７１１）。Ａ－パス表示フラグがセットされているならば、係数Ｃに対してＡ－パスが実行される（処理ブロック２７１７）。その後、処理ブロック２７１３に進む。Ａ－パス表示フラグがセットされていないときには、係数Ｃに対するＡ－パスを実行することなく処理ブロック２７１３へ進む。

【０１４７】処理ブロック２７１３において、係数Ｃが最終の係数であるか判定する。係数Ｃが最終の係数でなければ、処理ブロック２７１４に進み、次の係数が変数Ｃに割り当てられ、処理ブロック２７１９へ進む。しかし、係数Ｃが最終の係数であるならば、処理ブロック２７１５へ進みＢ－パスフラグ（do_B_flag）がセットされているか判定する。Ｂ－パスフラグがセットされているならば、ビットプレーンS_B－１をビットプレーンS_Bに設定し（処理ブロック２７１６）、処理ブロック２７１７へ進む。Ｂ－パス表示フラグがセットされていなければ、処理ブロック２７１７へ進む。処理ブロック２７１７では、Ａ－パスフラグがセットされているか判定する。セットされているならば、ビットプレーンS_A－１をビットプレーンS_AにトプレーンS_Aに設定し（処理ブロック２７１８）、プロセスは終了する。また、Ａ－パスフラグがセットされていなければ、プロセスは直ちに終了する。

【０１４８】実施例によっては、ある特定のビットプレーンの係数がＡ－グループであるかＢ－グループであるかを、フラグビットを使わずに判定することができる。

31

これは１係数あたり１ビットのメモリの節約になり、大きな画像の場合には意味があろう。フラグビットを使う代わりに、ＡＮＤ論理を用いて、マスクが係数と比較される。ＡＮＤの結果が０ならば、そのビットはＡ－グループであり、そうでなければＢ－グループである。これらマスクの例を８ビットプレーン分につき表７に示す（実施例説明の末尾）。なお、これらマスクは$2^{(bitplane+1)}$の２の補数である（符号ビット含まず）。

【０１４９】ＡーパスとＢーパスそれぞれに独自のマスクを割り当てることができるので、Ａーパスを必要なだけ、対応のＢーパスの前に実行することができる。１７ビットプレーンの場合、３個のＡーパスが実行され、次に１４個のＡーパスとＢーパスが同時に実行され、最後に２個のＢーパスが実行される。普通、Ａーパスのデシジョンのほうがより効率的に符号化できるので、初めに多数のＡーパスを実行すると、損失性圧縮の質を向上できる。

【０１５０】図５０及び図５１は、縮小フラグメモリ（後に詳細説明で述べる）を使用した本発明の符号化器の一実施例を示す。図５０において、符号化器の変換及びモデリングプロセスは、初めに入力データを獲得する（処理ブロック２８０１）。入力データを獲得した後、本発明は可逆ウエーブレットフィルタをかける（処理ブロック２８０２）。

【０１５１】次に、別のレベルの分割が必要であるか判定する（処理ブロック２８０３）。別レベルの分割を要するならば、処理ブロック２８０４に進み、直前の分割により得られたＬＬ係数に可逆ウエーブレットフィルタがかけられ、そして処理ブロック２８０３へ進む。このように、本発明によれば任意レベル数の分割を行なうことができる。

【０１５２】別レベルの分割が必要でないときには、処理ブロック２８０５へ進み、Ａーパス用ビットプレーンS_Aは最上位ビットプレーン（ｍａｘ）とされる。そして、Ｂーパス用ビットプレーンS_Bはその次位のビットプレーン（ｍａｘ－１）とされる（処理ブロック２８０６）。

【０１５３】次に、マスクM_Aが

【０１５４】

【数２４】

$$-2^{(S_A+1)}$$

【０１５５】に設定され（処理ブロック２８０７）、マスクM_Bが

【０１５６】

【数２５】

$$-2^{(S_B+1)}$$

【０１５７】に設定される（処理ブロック２８１４）。そして、Ａーパス用ビットプレーンS_Aを周波数ベースモデルで符号化すべきか否か判定する（処理ブロック２８０８）。ビットプレーンS_Aを周波数ベースモデルで

32

符号化すべきであれば、処理ブロック２８０９に進み、各係数の１ビットが周波数ベースモデルによりモデル化されてエントロピー符号化される。他方、ビットプレーンS_Aが周波数ベースモデルで符号化されるべきでないときには、処理ブロック２８１０に進み、各係数の１ビットが統合空間／周波数モデルによりモデル化されてエントロピー符号化される。

【０１５８】いずれの場合にも、次に処理ブロック２８１１に進み、ビットプレーンS_Aがゼロ以上であるか判定することにより、それが最後のビットプレーンであるか否かを表示する。ビットプレーンS_Aがゼロ以上であるならば、処理ブロック２８０８へループバックする。他方、ビットプレーンS_Aがゼロ以上でなければ、処理ブロック２８１２に進み、ビットプレーンS_BがＢーパスを施される最後のビットプレーンであるか判断するため、ビットプレーンS_Bがゼロ以上であるか判定する。ビットプレーンS_Bがゼロ以上であれば、処理ブロック２８０８から処理を繰り返す。しかし、ビットプレーンS_Bがゼロ以上でなければ、処理ブロック２８１３に進み、符号化データがチャネルへ転送されるかメモリに格納される。符号化データを格納または転送した後、本発明の符号化器の変換及びモデリングのプロセスは終了する。

【０１５９】図５２及び図５３は、縮小フラグメモリを使用した時の本発明の復号器の変換及びモデリングプロセスの別の実施例を示す。図５２において、本発明の復号器の変換及びモデリングプロセスは、まず符号化データを取り込む（処理ブロック２９０１）。この符号化データは、チャネルかメモリあるいはその他の伝送システムより受け取られるであろう。

【０１６０】符号化データが受け取られると、Ａーパス用ビットプレーンS_Aは最上位ビットプレーン（ｍａｘ）とされ（処理ブロック２９０３）、Ｂーパス用ビットプレーンS_Bはその次位のビットプレーン（ｍａｘ－１）とされる（処理ブロック２９０４）。各係数の値が初期値のゼロに設定される（処理ブロック２９０５）。次にマスクM_Bが

【０１６１】

【数２６】

$$-2^{(S_B+1)}$$

【０１６２】に設定され（処理ブロック２９０２）、またマスクM_Aが

【０１６３】

【数２７】

$$-2^{(S_A+1)}$$

【０１６４】に設定される（処理ブロック２９１５）。そして、ビットプレーンS_Aが周波数ベースモデルで復号されるべきか否かを判定する（処理ブロック２９０６）。ビットプレーンS_Aが周波数ベースモデルによって復号されるべきときには、処理ブロック２９０７へ進

み、各係数の１ビットが周波数ベースモデルでモデル化されてエントロピー復号される。ビットプレーンS_Aが周波数ベースモデルで復号されるべきでないならば、処理ブロック２９０８へ進み、各係数の１ビットが統合空間／周波数モデルでモデル化されてエントロピー復号される。

【０１６５】各係数のモデル化後、処理ブロック２９０９へ進み、ビットプレーンS_Aが最終のビットプレーンであるか判断するため、ビットプレーンS_Aがゼロ以上であるか判定する。ビットプレーンS_Aがゼロ以上ならば、処理ブロック２９０２へ進む。他方、ビットプレーンS_Aがゼロ以上でないと、Ｂ－パスのビットプレーンS_Bがゼロ以上であるか判定し（処理ブロック２９１０）、それによりビットプレーンS_Bが最後のＢ－パス用ビットプレーンであることを表示する。ビットプレーンS_Bがゼロ以上ならば、次の復号のため処理ブロック２９０２へ進む。他方、ビットプレーンS_Bがゼロ以上でないときには、最も粗い分割レベルによる係数に対して逆可逆フィルタがかけられる（処理ブロック２９１１）。次に、全レベルが逆フィルタ処理されたか判定する（処理ブロック２９１２）。違うときには、残っている最も粗い分割レベルに関する係数に対して逆可逆フィルタがかけられる（処理ブロック２９１３）。そして処理ブロック２９１２へ戻り、全レベルの逆フィルタ処理が済んだか判定する。

【０１６６】全レベルの逆フィルタ処理が済んだならば、処理ブロック２９１４に進み、復元データの格納または伝送が行なわれる。

【０１６７】図５４及び図５５は、各係数のモデリングプロセスの一実施例を示す。なお、図４８及び図４９と同様、図５４及び図５５のプロセスは図５０及び図５１並びに図５２及び図５３のモデリングステップの実現のために用い得るものである。図５４において、最初のプロセスはまず、Ａ－パスが必要で、かつ、S_Aがゼロ以上であるかを判定する（処理ブロック３００１）。その通りであるならば、Ａ－パスが実行されるべきであることを表示するフラグ（do_A_flag）がセットされ（処理ブロック３００４）、処理ブロック３００２へ進む。そうでないときには、フラグdo_A_flagはクリアされる（処理ブロック３００３）。

【０１６８】ビットプレーンS_AがビットプレーンS_Bより大きいときには、処理ブロック３００４に進み、Ａ－パスが実行されることを表示するためのフラグがセットされる。ビットプレーンS_AがビットプレーンS_Bより大きくないときには、処理ブロック３００３へ進み、Ａ－パスが生じることを表示するためのフラグはクリアされる。

【０１６９】処理ブロック３００３または３００４に続いて、処理ブロック３００２に進み、ビットプレーンS_Bがゼロ以上で、かつ、Ｂ－パスが必要であるかを判定

する。両ビットプレーンが等しくなければ、本発明はフラグ（do_B_flag）をクリアしてＢ－パスを発生させないようにし（処理ブロック３００５）、処理ブロック３００７へ進む。ビットプレーンS_BがビットプレーンS_Aと等しいときには、Ｂ－パスが実行されるべきことを表示するためフラグ（do_B_flag）がセットされ（処理ブロック３００６）、処理ブロック３００７へ進む。

【０１７０】処理ブロック３００７において、Ａ－パスフラグがセットされており、かつゼロツリーモデリングが行なわれるべきか判定する。フラグがＡ－パスが生じるべきで、かつゼロツリーモデリングが実行されるべきであることを表示しているときには、子を有する各係数に対して”確定／未確定”フラグが”未確定”状態に初期化され（処理ブロック３００８）、処理は処理ブロック３００９へ進む。他方、Ａ－パス表示フラグもゼロツリーモデリング表示もセットされていないときには、処理ブロック３００９へ直接進む。処理ブロック３００９において、最初の係数が変数Ｃに設定される。

【０１７１】最初の係数が変数Ｃに割り付けられたならばＢ－パス表示フラグがセットされているか判定する（処理ブロック３０１９）。Ｂ－パス表示フラグ（do_B_flag）がセットされているならば、本発明は係数Ｃに対してＢ－パスを実行し（処理ブロック３０１０）、処理ブロック３０１１へ進む。他方、Ｂ－パスフラグがセットされていないときには、Ｂ－パスはに対し実行されず、処理ブロック３０１１へ直接進む。

【０１７２】次にＡ－パス表示フラグがセットされているか判定する（処理ブロック３０１１）。Ａ－パス表示フラグがセットされているならば、Ａ－パスが係数Ｃに対して実行される（処理ブロック３０１２）。そして、処理ブロック３０１３へ進む。Ａ－パス表示フラグがセットされていないときには、係数Ｃに対しＡ－パスを実行することなく、処理ブロック３０１３へ進む。

【０１７３】処理ブロック３０１３において、係数Ｃが最終の係数であるか判定する。係数Ｃが最終の係数でなければ、処理ブロック３０１４へ進み、次の係数が変数Ｃに割り付けられ、処理ブロック３０１９へ進む。しかし、係数Ｃが最終の係数であるならば、処理ブロック３０１５へ進み、Ｂ－パスフラグ（do_B_flag）がセットされているか判定する。Ｂ－パスフラグがセットされているならば、ビットプレーンS_B－１がビットプレーンS_Bとされ（処理ブロック３０１６）、処理ブロック３０１７へ進む。Ｂ－パス表示フラグがセットされていないときには、処理ブロック３０１７へ進む。処理ブロック３０１７では、Ａ－パスフラグがセットされているか判定する。セットされているならば、ビットプレーンS_A－１がビットプレーンS_Aとされ（処理ブロック３０１８）、処理は終了する。また、Ａ－パスフラグがセットされていなければ、処理が直ちに終了する。

【０１７４】＜係数ツリー＞ピラミッドシステムでは、

35

ツリー構造を用いて係数を複数の組みにグループ分けできる。各ツリーのルートは純粋に１つの低域通過係数である。図１４は、変換画像の純粋に１つの低域通過係数に関するツリー構造を示す。画像のような二次元信号の場合、ツリーのルートは３個の”子”を持ち、ノード以外のノードはそれぞれ４個の子を持つ。ツリー階層構造は二次元信号に限定されない。例えば、一次元信号の場合、ルートは１個の子を持ち、ルート以外のノードはそれぞれ２個の子を持つ。一次元及び二次元のケースから、より多次元のケースが分かる。

【０１７５】ツリー構造は、図９乃至図１１に示したフィルタの動作からも明らかである。フィルタのペア群の作用とサブサンプリングにより、前述の係数は関係付けられる。

【０１７６】本発明においては、係数が符号－絶対値形式にされた後、コンテキストモデルが複数の符号化法のどれを、係数をさらに符号化するために用いるべきか判断する。ゼロツリー符号化（zerotree coding）のような周波数ベース符号化方式は、一定のサブバンド分割に関連した特定の閾値に対する有意データを効率的に符号化する。関連したサブバンド分割の孤立した単一の係数の有意（singificance）または無意（insignificance）を表現するシンボルを用いることに加え、全部の子が無意である（全部の子が一定の閾値以下の絶対値を持つ）無意の親のエントリーは一つにまとめられ、一緒に符号化される。これらのツリーはゼロツリー（zerotree）と呼ばれることがある。これらの無意なツリーは、ゼロツリールート（zerotree root）と呼ばれることのある一つの専用シンボルで符号化される。しかしながら、有意な子孫が存在するならば、無意な係数のエントリーは”孤立ゼロ”（isolated zero）用シンボルを用いて符号化される。かくして、１つのツリーは、係数の符号（sign）が未だ符号化されていないデシジョンについて、４つのシンボル（積極的有意、消極的有意、孤立ゼロまたはゼロツリールート）を用いて符号化される。

【０１７７】周波数ベース符号化は、圧縮システムで特に有用である。というのは、無意なツリーの一括符号化が、少数の親係数により多数の子孫係数の無意を予測可能にするからである。ツリー中の子孫係数に関連したエントリーはルートから予測できるので、それらの無意を符号化するために余分なシンボルは必要とされない。ツリー全体の無意は非常に低コストで符号化される。よって、上位のビットプレーンは、殆ど、その多くがゼロツリールートでも孤立ゼロでもない（すなわち符号化される必要のない無意なツリーの子である）無意な係数からできている。

【０１７８】Ｓｈａｐｉｒｏは、Ｚｅｒｏｔｒｅｅと呼ばれる周波数ベースモデルを米国特許第５，３２１，７７６号で開示している。Ｓｈａｐｉｒｏの方法では、主リストと副リストという２つのリストを用いて係数全部

36

を記憶する。シグニフィカンス（significance）レベル毎に、主パスと副パスという２つのパスが作られる。一実施例では、本発明の周波数ベースモデルはＺｅｒｏｔｒｅｅである。

【０１７９】別の実施例においては、Ｚｅｒｏｔｒｅｅ（Shapiroによって述べられた）に類似した周波数ベースモデルが用いられる。複数のリストを使うのではなく、リストが１つだけ用いられ、そのリストの各エレメントは２グループの中の一方のメンバーである旨マークされる。係数をＡ－グループとＢ－グループに分類することは、Ｓｈａｐｉｒｏが主リストと副リストを用いて行なった分類に相当する。Ｓｈａｐｉｒｏの複数リストを使う方法は、副リスト中の係数の順序付けのフレキシブリティを大きくできるが、ソフト／ハードの複雑さが増加するという犠牲を伴う。本実施例の単一リスト・ゼロツリー（Single List Zerotree）法は、Ａ－パスとＢ－パスという２つのパスを使う。これら２つのパスはそれぞれ、Ｓｈａｐｉｒｏの主パスと副パスに相当する。この単一リスト・ゼロツリーモデルについて、以下に述べる。

【０１８０】本発明の符号化システムは符号－絶対値形式の係数の１つのリストをメモリ内に作る。このリストの各エレメントは、該エレメントが”Ａグループまたは Ｂ－グループ”のいずれのメンバーであるかを表示する１ビットのラベルを持っている。あるステージの初めで、有意であると判明していない係数はＡ－グループに属するとしてラベル付けされる。前のより大きな閾値に対して有意であると前に判明している係数は、Ｂ－グループであるとラベル付けされる。リストは、係数を符号化の処理がなされる順に格納する。文字通り最初のステージの開始時点では、どの係数も有意であるとは確定していないので、全ての係数がＡ－グループのメンバーとしてラベル付けされている。係数が有意または無意であると判定された時に、それら係数のエントリーが元のＡ－グループ表示からＢ－グループ表示へ変更される。このリストは閾値を次第に細かくして順次更新される。すなわち、リストのマルチパスが生じる。

【０１８１】一実施例では、Ｂ－グループに対応する２元イベントは０次マルコフコンテキストモデルの下でバイナリ算術符号化される。Ａ－グループの係数に対応する４元イベントも、０次マルコフコンテキストモデルの下で符号化される。

【０１８２】子がその親より先にモデル化されることがないよう、本発明のリスト内の係数の順序はツリー構造を保存する。したがって、ツリー構造を保存する、ある順序付けが決まり、それが一貫して使われる。一実施例では、係数は最初に使用された記憶ロケーションより順番にメモリに格納される。別の実施例では、連結リスト（linked list）が用いられるかもしれない。

【０１８３】一実施例では、係数はビット・シグニフィ

カンスもしくはビットプレーン埋め込み方式で符号化される。係数は最上位のビットプレーンから最下位のビットプレーンへと符号化されるので、データのビットプレーン数を確定しなければならない。本発明では、係数値の絶対値の上限を、データから計算するか、画像の濃さ（depth）とフィルタ係数より得ることによって、ビットプレーン数の確定がなされる。一例を挙げれば、上限が１４９のときは、８ビットのシグニフィカンスもしくは８つのビットプレーンがある。

【０１８４】図１５及び図１６は本発明の単一リスト・ゼロツリー符号化プロセスを示す。一実施例では、図１５及び図１６に示すプロセスは図４８及び図４９に示すモデリングプロセスで用いられる。図１５において、プロセスは初めに係数Ｃのグループフラグが”Ａ－グループ”に設定されているか調べる（処理ブロック３２２１）。”Ａ－グループ”に設定されていないときは、プロセスは終わる。他方、係数Ｃのグループフラグが”Ａ－グループ”に設定されているときは、処理ブロック３２２２に進み、Ｃ係数の”確定／未確定”フラグが”未確定”に設定されているか判定する。係数Ｃの”確定／未確定”フラグが”未確定”に設定されていなければ、プロセスは終了する。しかし、係数Ｃの”確定／未確定”フラグが”未確定”に設定されているならば、処理ブロック３２０３に進み、係数ＣのビットＳA が１であるか判定する。

【０１８５】係数ＣのビットＳA が１でなければ、処理ブロック３２０７（図１６）へ進む。他方、係数ＣのビットＳA が１のときには、処理ブロック３２０４へ進み、係数Ｃの符号（sign）が正であるか判定する。係数Ｃの符号が正でないならば、デシジョンは”Ａ－グループ”コンテキストの”消極的有意”として符号化され（処理ブロック３２０５）、処理ブロック３２２９へ進む。係数Ｃの符号が正であるならば、デシジョンは”Ａ－グループ”コンテキストの”積極的有意”として符号化され（処理ブロック３２０６）、処理ブロック３２２９へ進む。処理ブロック３２２９ではＣのグループフラグは”Ｂ－グループ”に設定される。

【０１８６】処理ブロック３２０７で、係数Ｃの全ての子孫（子）についてビットＳA が０であるか判定する。ビットＳA が０でないときには、テシジョンは”Ａ－グループ”コンテキストの”有意な子のある無意”（０１）として符号化され（処理ブロック３２０８）、プロセスは終了する。他方、係数Ｃの全部の子孫（子）についてビットＳA が０であるときには、”Ａ－グループ”コンテキストの”ゼロルート”（００）として符号化される（処理ブロック３２０９）。それから、係数Ｃの子孫全部に対する”確定／未確定”フラグが”確定”に設定され（処理ブロック３２２１）、処理は終了する。

【０１８７】別の実施例では、プロセスの終了判定は、所望の圧縮率に達したか否かでなされる。

【０１８８】一実施例では、Ｂ－パスから得られる２元イベントは、０次マルコフ情報源コンテキストモデルの下でエントロピー符号化される。Ａ－パスより得られる２ビットのアルファベット（サイズは４）も、０次マルコフ情報源コンテキストモデル下で、４元（サイズ４のアルファベット）算術符号化器により符号化される。図１７及び図１８は、縮小フラグメモリを使う本発明の単一リストゼロツリー符号化プロセスの別の実施例を示す。一実施例では、図１７及び図１８のプロセスが図５４及び図５５のプロセスにおけるＡ－パスとして使用される。図１７において、プロセスは初めに係数ＣとマスクMA とを論理積した結果が０であるか判定する（処理ブロック３２０１）。０でなければ、プロセスは終了する。他方、係数ＣをマスクMA と論理積した結果が０ならば、処理ブロック３２０２に進み、係数Ｃの親の”確定／未確定”フラグが”未確定”に設定されているか判定する。係数Ｃの親の”確定／未確定”フラグが”未確定”に設定されていないときは、プロセスは終了する。しかし、係数Ｃの親の”確定／未確定”フラグが”未確定”に設定されているときには、処理ブロック３２０３に進み、係数ＣのビットＳA が１であるか判定する。

【０１８９】係数ＣのビットＳA が１でなければ、処理ブロック３２０７へ進む。他方、係数ＣのビットＳA が１であるならば、処理ブロック３２０４に進み、係数Ｃの符号（sign）が正であるか判定する。係数Ｃの符号が正でなければ、デシジョンは”Ａ－グループ”コンテキストの”消極的有意”で符号化され（処理ブロック３２０５）、プロセスは終了する。係数Ｃの符号が正であるならば、デシジョンは”Ａ－グループ”コンテキストの”積極的有意”として符号化され（処理ブロック３２０６）、プロセスは終了する。一実施例では、４元符号化器が用いられ、４元デシジョンが１つのコンテキストで符号化される。別の実施例では、バイナリ符号化器が使用され、３つのコンテキストが用いられる（例えば、この３つのコンテキストは、デシジョンの第１ビット、第１ビットが０のときの第２ビット、及び第１ビットが１のときの第２ビット、である）。

【０１９０】処理ブロック３２０７では、係数Ｃの子孫（子）全てにつきビットＳA が０であるか判定する。ビットＳA が０でなければ、デシジョンは”Ａ－グループ”コンテキストの”有意の子がある無意”，”孤立ゼロ”（０１）として符号化され（処理ブロック３２０８）、プロセスは終了する。他方、係数Ｃの子孫（子）全てにつきビットＳA が０であるならば、デシジョンは”Ａ－グループ”コンテキストの”ゼロツリー・ルート”（００）として符号化される（処理ブロック３２０９）。そして、係数Ｃの”確定／未確定”フラグが”確定”に設定される（処理ブロック３２１０）。続いて、係数の子孫（同様に子孫を持つ）の全部に対する”確定／未確定”フラグが”確定”に設定され（処理ブロック

３２１１）、プロセスは終わる。

【０１９１】＜復号ステップ＞本発明においては、復号は符号化と密接行進的に行なわれる。

【０１９２】図１９はゼロツリー水平符号化（zerotree horizon coding）プロセスのためのＡ－パスプロセスの一実施例を示し、これは図４８及び図４９のプロセスと共に用い得る。図１９において、プロセスは初めに係数Ｃのグループフラグが”Ａ－グループ”に設定されているか判定する（処理ブロック３５２１）。”Ａ－グループ”に設定されていなければ、プロセスは終了する。しかし、”Ａ－グループ”に設定されているならば、処理ブロック３５２８に進み、係数Ｃの”確定／未確定”フラグが”未確定”に設定されているか判定する。”未確定”に設定されていなければ、プロセスは終了する。”未確定”に設定されているならば、処理ブロック３５０２に進み、３元デシジョンはＡ－グループコンテキストで復号される。

【０１９３】次に、デシジョンが”積極的有意”であるか判定する（処理ブロック３５０３）。デシジョンが”積極的有意”であるならば、係数の符号が正に設定され（処理ブロック３５０５）、係数の絶対値が

【０１９４】

【数２８】

$$2^{S_A}$$

【０１９５】に設定され（処理ブロック３５０７）、係数Ｃのグループフラグが”Ｂ－グループ”に設定され（処理ブロック３５４１）、処理は終了する。

【０１９６】デシジョンが”積極的有意”でないときは（処理ブロック３５０３）、デシジョンが”消極的有意”であるか判定する（処理ブロック３５０４）。デシジョンが”消極的有意”でないときには、処理ブロック３５１０に進み、デシジョンが”ゼロツリー・ルート”であるか判定する。デシジョンが”ゼロツリー・ルート”でなければ、プロセスは終わる。デシジョンが”ゼロツリー・ルート”ならば、係数Ｃの子孫全部の”確定／未確定”フラグが”未確定”に設定され（処理ブロック３５３１）、プロセスは終了する。

【０１９７】しかし、処理ブロック３５０４の判定でデシジョンが”消極的有意”と判断されたときには、係数Ｃの符号が負に設定され（処理ブロック３５０６）、係数Ｃの絶対値が

【０１９８】

【数２９】

$$2^{S_A}$$

【０１９９】に設定され（処理ブロック３５０７）、係数Ｃのグループフラグがグループに設定され（処理ブロック３５４１）、プロセスは終了する。

【０２００】図２０は縮小フラグメモリを用いるゼロツリー水平復号プロセスのためのＡ－パスプロセスの他の実施例を示し、これは図５４及び図５５に述べたプロセスにおいて用いることができる。図２０において、プロセスは初めに係数ＣとマスクＭＡの論理積結果が０であるか判定する（処理ブロック３５０１）。０でなければ、プロセスは終了する。しかし、係数ＣとマスクＭＡを論理積した結果が０ならば、処理ブロック３５０８へ進み、係数の”確定／未確定”フラグが”未確定”であるか判定する。”未確定”でなければ、処理は終了する。”未確定”であれば、処理ブロック３５０２に進み、３元デシジョンはＡ－グループコンテキストで復号される。

【０２０１】次に、デシジョンが”積極的有意”であるか判定する（処理ブロック３５０３）。デシジョンが”積極的有意”であるならば、係数の符号が正に設定され（処理ブロック３５０５）、係数の絶対値が

【０２０２】

【数３０】

$$2^{S_A}$$

【０２０３】に設定され（処理ブロック３５０７）、プロセスは終了する。

【０２０４】デシジョンが”積極的有意”でなければ、それが”消極的有意”であるか判定する（処理ブロック３５０４）。”消極的有意”でなければ、処理ブロック３５０９に進み、デシジョンが”ゼロツリー・ルート”であるか判定する。デシジョンがゼロツリー・ルートでなければ、プロセスは終了する。デシジョンがゼロツリー・ルートであるならば、係数Ｃの”確定／未確定”フラグが”確定”に設定され（処理ブロック３５１０）、係数Ｃの全子孫（これも同様に子孫を持つ）に対する”確定／未確定”フラグが”確定”に設定され（処理ブロック３５１１）、プロセスは終了する。

【０２０５】しかしながら、処理ブロック３５０４のテストでデシジョンが”消極的有意”であると判定したときには、係数Ｃの符号は負に設定され（処理ブロック３５０６）、係数Ｃの絶対値は

【０２０６】

【数３１】

$$2^{S_A}$$

【０２０７】に設定され（処理ブロック３５０７）、プロセスは終了する。

【０２０８】Ｓｈａｐｉｒｏは４元デシジョンを使ってツリーを記述する方法を選択したが、その代案が存在する。ツリーのルートを符号化する時にツリー全体の特性をより詳細に記述するため、拡大したアルファベットを使用してもよい。一実施例では、以下の６元デシジョンの集合が用いられる。

【０２０９】・無意の子を持つ無意（ゼロツリー・ルー

ト）
・少なくとも１つの有意な子を持つ無意
・有意、積極的かつ全てのが非消極的（non-negative）
・有意、積極的かつ少なくとも１つの子が消極的
・有意、消極的かつ全ての子が非積極的（non-positive）
・有意、消極的かつ少なくとも１つの子が積極的
この実施例では、ツリー全体について、無意に加え符号（sign）情報も予測される。別の実施例では、他の符号（sign）制限または絶対値制限のあるツリーが予測可能である。択一的予測器が、テクスチャあるいはマルチ解像度のフィーチャを表現する場合に特に有効かもしれない。アルファベットの拡大とともに、高次のマルコフコンテキスト（後述）を使用するのが有効かもしれない。

【０２１０】＜マルチパスリストベース（Multipass List Based）統合空間／周波数埋め込みモデリング＞本発明において、本明細書に開示された水平順序（horizon order）モデリングのような周波数埋め込み符号化（frequency embedded coding）は、Ａ－グループの係数に対応する３元イベントを符号化する。水平符号化では、符号化ステップに先行する初期化は全て、周波数ベースのシステムと同一である。一実施例では、バイナリエントロピー符号化は、”Ａ－グループ　絶対値”、”Ａ－グループ　符号（sign）”、及び”Ｂ－グループ”という３つのコンテキストで行なわれる。

【０２１１】図２１は本発明の単一リスト水平符号化プロセスのためのＡ－パスの一実施例のフローチャートである。このプロセスは図４８及び図４９のプロセスで使用し得る。図２１において、このＡ－パスプロセスは初めに係数Ｃのグループフラグが”Ａ－グループ”であるか判定する（処理ブロック３１１１）。”Ａ－グループ”でなければ、プロセスは終了する。係数Ｃのグループフラグが”Ａ－グループ”に設定されているときは、処理ブロック３１０２に進み係数ＣのビットＳＡが１であるか判定する。係数ＣのビットＳＡが１でなければ、デシジョンは”Ａ－グループ”コンテキストの無意（０）として符号化され（処理ブロック３１０３）、プロセスは終了する。係数ＣのビットＳＡが１ならば、処理ブロック３１０４へ進み係数Ｃの符号（sign）が正であるか判定する。係数Ｃの符号が正ならば、デシジョンは”Ａ－グループ”コンテキストの”積極的有意”（１０）として符号化され（処理ブロック３１０６）、処理ブロック３１１７へ進む。他方、係数Ｃの符号が正でなければ、デシジョンは”Ａ－グループ”コンテキストの”消極的有意”（１１）として符号化され（処理ブロック３１０５）、処理ブロック３１１７へ進む。処理ブロック３１１７で、係数Ｃのグループフラグが”Ｂ－グループ”に設定される。

【０２１２】図２２は、縮小フラグメモリを使用する本発明の単一リスト水平符号化プロセスのためのＡ－パス

の他の実施例のフローチャートである。このプロセスは図５４及び図５５のプロセスで用い得る。図２２において、このＡ－パスプロセスは初めに係数ＣをマスクＭＡと論理積した結果が０であるか判定する（処理ブロック３１０１）。０でなければ、プロセスは終了する。係数ＣのマスクＭＡの論理積結果が０であるときは、処理ブロック３１０２に進み係数ＣのＳＡが１であるか判定する。係数ＣのＳＡが１でなければ、デシジョンは”Ａ－グループ”コンテキストの無意（０）として符号化され（処理ブロック３１０８）、プロセスは終了する。係数ＣのＳＡが１であるならば、処理ブロック３１０４へ進み係数Ｃの符号（sign）が正であるか判定する。係数Ｃの符号が正であるならば、デシジョンは”Ａ－グループ”コンテキストの”積極的有意”（１０）として符号化され、プロセスは終了する。他方、係数Ｃの符号が正でなければ、デシジョンは”Ａ－グループ”コンテキストの”消極的有意”（１１）として符号化され（処理ブロック３１０５）、プロセスは終了する。

【０２１３】＜復号ステップ＞図２３は本発明の単一リスト水平復号プロセスのためのＡ－パスプロセスの一実施例を示し、図４８及び図４９のプロセスに利用し得る。図２３において、このプロセスは初めに係数Ｃのグループフラグが”Ａ－グループ”に設定されているか判定する（処理ブロック３４１１）。”Ａ－グループ”に設定されていなければ、プロセスは終了する。しかし、係数Ｃのグループフラグが”Ａ－グループ”に設定されているならば、処理ブロック３４０２に進み３元デシジョンはＡ－グループコンテキストで復号される。

【０２１４】次に、デシジョンが”積極的有意”であるか判定する（処理ブロック３４０３）。デシジョンが”積極的有意”であるならば、係数Ｃの符号（sign）が正に設定され（処理ブロック３４０５）、係数の絶対値が

【０２１５】

【数３２】

$$2^{S_A}$$

【０２１６】に設定され（処理ブロック３４０７）、係数Ｃのグループフラグが”Ｂ－グループ”に設定され（処理ブロック３４１８）、プロセスは終了する。

【０２１７】デシジョンが”積極的有意”でなければ、デジジョンが”消極的有意”であるか判定する（処理ブロック３４０４）。デシジョンが”消極的有意”でなければ、プロセスは終了する。しかし、デシジョンが”消極的有意”であるならば、係数Ｃの符号が負に設定され（処理ブロック３４０６）、係数Ｃの絶対値が

【０２１８】

【数３３】

$$2^{S_A}$$

【０２１９】に設定され（処理ブロック３４０７）、係

数Cのグループフラグが”Bーグループ”に設定され（処理ブロック３４１８）、プロセスは終了する。

【０２２０】図２４は縮小フラグメモリを使用する単一リスト水平復号プロセスのためのAーパスプロセスの他の実施例を示し、これは図５４及び図５５のプロセスにおいて用い得る。図２４において、プロセスは初めに係数CとマスクMA の論理積結果が０であるか判定する（処理ブロック３４０１）。０でなければ、プロセスは終了する。しかし、係数CとマスクMA の論理積結果が０ならば、処理ブロック３４０２に進み３元デシジョンはAーグループコンテキストで復号される。

【０２２１】次にデシジョンが”積極的有意”であるか判定する（処理ブロック３４０３）。デシジョンが”積極的有意”ならば、係数Cの符号（sign）が正に設定され（処理ブロック３４０５）、係数の絶対値が

【０２２２】

【数３４】

$$2^{S_A}$$

【０２２３】に設定され（処理ブロック３４０７）、プロセスは終了する。

【０２２４】デシジョンが”積極的有意”でなければ、デシジョンが”消極的有意”であるか判定する（処理ブロック３４０４）。デジジョンが”消極的有意”でなければプロセスは終了する。しかし、デシジョンが”消極的有意”であるならば、係数Cの符号（sign）が負に設定され（処理ブロック３４０６）、係数Cの絶対値が

【０２２５】

【数３５】

$$2^{S_A}$$

【０２２６】に設定され（処理ブロック３４０７）、プロセスは終わる。

【０２２７】＜ゼロツリー符号化／復号及び水平符号化／復号のためのBーパス＞一実施例においては、本発明のゼロツリー符号化／復号と水平符号化／復号のためのBーパスプロセスは同一である。符号化プロセスと復号プロセスのためのBーパスアルゴリズムの実施例が図２５及び図２６と図２７及び図２８にそれぞれ示されている。

【０２２８】図２５はゼロツリー符号化及び単一リスト水平符号化のプロセスのために部分的に用いられるBーパスプロセスの一実施例を示し、図４８及び図４９のプロセスに利用し得る。図２５において、プロセスは最初に係数Cのグループフラグが”Bーグループ”に設定されているか判定する（処理ブロック３３１１）。”Bーグループ”に設定されていなければ、プロセスは終了する。他方、グループフラグが”Bーグループ”に設定されているならば、処理ブロック３３０２に進み係数CのビットSB が”１”であるか判定する。係数CのビットSB が”１”でなければ、デシジョンは”Bーグループ”コンテキストの”０”として符号化され（処理ブロック３３０３）、プロセスは終了する。係数CのビットSB が”１”であるならば、デシジョンは”Bーグループ”コンテキストの”１”として符号化され（処理ブロック３３０４）、プロセスは終了する。

【０２２９】図２６はゼロツリー符号化及び単一リスト水平符号化のプロセスに部分的に用いられる、縮小フラグメモリを使うBーパスプロセスの他の実施例を示し、図５４及び図５５のプロセスに使用し得る。図２６において、プロセスは最初にマスクMB と係数Cの論理積結果が０でないか判定する（処理ブロック３３０１）。０であるならば、プロセスは終了する。他方、マスクMB と係数Cの論理積結果が０でなければ、処理ブロック３３０２に進み係数CのビットSB が”１”であるか判定する。係数CのビットSB が”１”でなければ、デシジョンは”Bーグループ”コンテキストの”０”として符号化され（処理ブロック３３０３）、プロセスは終了する。係数CのビットSB が”１”であるならば、デシジョンは”Bーグループ”コンテキストの”１”として符号化され（処理ブロック３３０４）、プロセスは終了する。

【０２３０】図２７は本発明のBーパス復号の一実施例を示し、図４８及び図４９のプロセスに利用し得る。図２７において、最初に係数Cのグループフラグが”Bーグループ”に設定されているか判定する（処理ブロック３６１１）。そのように設定されていなければ、プロセスは終了する。しかし、係数Cのグループフラグが”Bーグループ”に設定されているならば、デシジョンは”Bーグループ”コンテキストで復号される（処理ブロック３６０２）。次に、デシジョンが”１”であるか判定する（処理ブロック３６０３）。デシジョンが”１”でなければプロセスは終了する。デシジョンが”１”ならば、係数CのビットSB がセットされ（処理ブロック３６０４）、プロセスは終了する。

【０２３１】図２８は縮小フラグメモリを使用する本発明のBーパス復号の他の実施例を示し、図５４及び図５５のプロセスにおいて使用できる。図２８において、最初に係数CとマスクMB との論理積の結果が非０か判定する（処理ブロック３６０１）。係数CとマスクMB との論理積の結果が０ならば、プロセスは終了する。しかし、係数CをマスクMB と論理積した結果が非０ならば、デシジョンは”Bーグループ”コンテキストで復号される（処理ブロック３６０２）。そして、デシジョンが”１”であるか判定する（処理ブロック３６０３）。デシジョンが”１”でなければ、プロセスは終了する。デシジョンが”１”であれば、係数CのビットSB がセットされ（処理ブロック３６０４）、プロセスは終了する。

【０２３２】本発明は、ゼロツリー順序号化と水平順序

符号化の組合せを用いて、可逆ウエーブレットにより生成された係数のビット・シグニフィカンス符号化を行なう。なお、Ａ－グループとＢ－グループの両方並びに”Ａ”パスと”Ｂ”パスに対応する３元と２元のイベントを使うことは、ゼロツリー順序付けの利用から水平順序付けの利用への切り替えが任意のＡパスの終わりでなされるという点で、非常に重要である。これは、下位ビットでのゼロツリー順序付けに伴う予測の非効率を補う。したがって、本発明においては、システムは初めに上位ビットデータをゼロツリー符号化し、リストのある回数のパスの後に、つまり、いくつかのビットプレーンが符号化された後に、本発明の符号化器は水平符号化によって残りのデータを符号化するように切り替わる。そのパスの回数は、統計的に選んでもよいし、あるいは、ゼロツリー順序付け符号化ブロックの性能をモニタして適応的に選んでもよい。

【０２３３】＜コンテキストモデルの代案＞一実施例では、５個のバイナリコンテキストビン（bin）が用いられる。これは、１０２４個より若干多いコンテキストを用いるＪＢＩＧのような他のシステムに比べると少ない。もっと多くのコンテキストを用いれば、圧縮率は向上するであろう。デシジョンは、空間的位置、レベル、及び／またはビット位置により条件付けられてもよい。デシジョンは、現在データと空間的位置、レベル及び／またはビット位置が接近した、前に符号化されたデータにより条件付けられてもよい。一般的に、前述の０次マルコフコンテキストは、より高次のマルコフコンテキストによって置き換えてもよい。

【０２３４】いくつか例を挙げるならば、次のとおりである。各仮数（いくつかの実施例におけるＢ－グループデータ）の最上位の（したがって最も容易に予測される）ビットは、その他のビットとは異なったコンテキストを用いてもよい。有意／非有意のデシジョンは、同一変換レベルの空間的に接近した前の係数に対して下された同様のデシジョンにより条件付けされてもよい。同様に、有意な係数の符号ビットは、同一レベルの空間的に接近した前の係数の符号ビット、あるいは親の係数の符号ビットにより条件付けられてもよい。

【０２３５】コンテキストモデルの改良は、空間的構造またはマルチ解像度構造を有する画像を圧縮する場合に特に重要である。線図形またはテキストの濃淡画像は、そのような種類の構造の両方を持つ画像の一例である。コンテキストモデルの改良は、ある指定されたピーク誤差で圧縮及び伸長されなければならないファイルの圧縮にも重要である。

【０２３６】＜他の実施例＞本発明は、ハードウエア及び／またはソフトウエアで実現し得る。ハードウエアにより本発明を実施するには、ウエーブレットフィルタ、同フィルタにデータを供給するためのメモリ／データフロー管理機能、本発明の埋め込み符号化を制御するため

のコンテキストモデル、同コンテキストモデルにデータを提供するためのメモリ／データフロー管理機能、及びバイナリエントロピー符号化器を実現する必要がある。

【０２３７】＜ウエーブレットフィルタ＞本発明のフォワード・ウエーブレットフィルタの一実施例が図２９に示されている。図２９に示したウエーブレットフィルタは、x(2)～x(5)として示した１６ビットの２の補数の入力画素を４個取り込む。

【０２３８】図２９において、２タップ”１１”低域通過フィルタは１個の１６ビット加算器１００１を用いる。その出力はそれぞれＳとＤと呼ばれる。この加算器の出力（Ｓ）は、１ビットシフトブロック１００３により１６ビットに丸められる。この１ビットシフトブロック１００３は、１７ビット入力を１ビット右シフトすることにより２で割る働きをする。

【０２３９】６タップ”-1 -1 8 -8 1 1”高域通過フィルタは、$-S_0+4D_1+S_2$ という計算を必要とする。S_2-S_0 の計算は、１ビットシフトブロック１００３の出力及び$Y_0(0)$を受け取る１６ビット減算器１００５により求められる。$4D_1$項は、減算器１００２と２ビットシフトブロック１００４を用いて計算される。１６ビット減算器１００２の出力は、２ビット左シフトされることにより４倍される。２ビットシフトブロック１００４の$4D_1$ 出力と減算器１００５の出力の加算が２０ビット加算器１００６により行なわれる。この最後の加算器の出力は、２ビットシフトブロック１００７によって１８ビットに丸められる。この２ビットシフトブロック１００７は、その２０ビット入力を２ビット右シフトすることにより、４で割る働きをする。

【０２４０】このように、必要とされる演算ハードウエア全体（中間結果を格納するためのレジスタは数にいれない）は、

・１６ビット加算器　１個
・１６ビット減算器　２個
・１９ビット加算器　１個

である。なお、シフトは配線により行なわれるので論理は不要である。

【０２４１】他の実施例では、Ｎビットサイズの入力の場合、１個のＮビット加算器、２個のＮビット減算器及び１個の（Ｎ＋３）ビット加算器が使われることになろう。

【０２４２】これらの加算器／減算器はハードウエアコストが非常に低いので、希望するならばフィルタの並列構成も利用できる。

【０２４３】あるいは、x(3)とx(2)の減算の代わりに、x(4)－x(5)を計算し、これを次のシフトまたはフィルタ処理のためのx(2)－x(3)として必要になるまでセーブしてもよい。このフォワードフィルタ（及び下に述べる逆フィルタ）のいずれも、より高いスループットを得るためパイプライン化してもよい。

47

【０２４４】逆ウエブレットフィルタが図３０に示されている。$Y_0(0)$，$Y_0(2)$入力は減算器１１０１により減算される。この減算の結果は２ビットシフトブロック１１０２で右に２ビットシフトされる。これは事実上、減算器１１０１の出力を４で割算することである。この２ビットシフトブロック１１０２の出力と入力$Y_1(0)$との間で減算が行なわれる。入力$Y_0(1)$は、１ビットシフトブロック１１０３により左に１ビットシフトされることにより、２倍される。$Y_0(1)$が１ビットシフト（２倍）された後、そのシフト後の値のＬＳＢは減算器１１０４の出力から得られたＬＳＢであり、１ビットシフトブロック１１０３の１６ビット出力と結合されて加算器１１０５と減算器１１０６の入力となる。加算器１１０５及び減算器１１０６のもう一方の入力は、減算器１１０４の出力である。加算器１１０５及び減算器１１０６の出力はその後、クリッピングを施される。

【０２４５】２つのクリップ操作の１つを選んで使用できる。いずれの場合でも、２０ビットの値が１ビットシフトされ（２で割られ）、１９ビット値とされる。非損失性圧縮だけを実行するシステムの場合、下位１６ビットを出力してよい（残り３ビットは無視してよい）。損失性システム（または損失性／非損失性システム）では、１９ビット値は、負ならば０に設定され、$2^{16}-1$を超えるときには$2^{16}-1$に設定され、それ以外ならば下位１６ビットを出力してよい。

【０２４６】Ｎビットサイズの入力の場合、１個のＮビット減算器、１個の（Ｎ＋２）ビット減算器、１個の（Ｎ＋３）ビット加算器及び１個の（Ｎ＋３）ビット減算器が使われることになろう。そして、クリップユニットはＮビットを出力する。

【０２４７】＜メモリの使い方＞本発明のウエーブレットフィルタのためのメモリ及びデータフロー管理に関しては、１枚のフルフレームがメモリにぴったり入る画像の場合、メモリ／データフロー管理は難しい問題ではない。１０２４×１０２４×１６ビットの医用画像（すなわちサイズが２Ｍバイト）の場合でさえも、１つのフルフレームバッファを要求することは多くの用途にとって合理的である。より大きい画像（例えば、Ａ４，４００ＤＰＩ，４色の画像は約５０Ｍバイトの大きさである）については、限られた量のラインバッファメモリを用いてウエーブレット変換を行なうのが望ましい。

【０２４８】なお、本発明が１パスシステムを実現するのにフルフレームバッファは必要でない。このため、必要とされるメモリは１００分のいくつかになる（大きな画像のためのフルフレームバッファを使用する場合に比べ）。本発明の１パスシステムは後述する。

【０２４９】フィルタメモリに格納されるデータは、埋め込み符号化及びバイナリエントロピー符号化される係数の系列である。埋め込み符号化は、周波数ベース符号化または水平符号化の利用を調整し、かつデータを適当な順序で供給するため、あるコンテキストモデルを用いる。このコンテキストモデルは、メモリ管理手順と連関して動作する。フルフレームメモリを持つシステムでは、データを適当な順序で供給することは難しくない。フルフレームバッファを持たないシステムの場合、本発明の１パスの実施例の変換データ管理手順（後述）は、コンテキストモデルが１つのツリー分の係数だけバッファすればよいように係数をコンテキストモデルに供給する。１パスの周波数ベースコンテキストモデル及び１パスの統合空間／周波数コンテキストモデルは、一度に１つのツリーを処理する。

48

【０２５０】本発明の埋め込み操作の結果は、本発明の周波数ベースモデリング機構及び本発明の統合空間／周波数モデリング機構よりビットストリームを生成させる。これらのビットストリームは、次にバイナリエントロピー符号化器により符号化される。

【０２５１】フルフレームバッファを持つシステムでは、どのようなバイナリエントロピー符号化器（あるいは他の適当な符号化器）を用いてもよい。フルフレームバッファを持たないシステムでは、複数の独立した符号化器を使用するか、あるいは１つの符号化器が複数の独立した符号化器をシミュレートできるか、のいずれかでなければならない。また、独立した複数の符号化器の出力を管理するためにメモリまたはチャネルの管理が必要となる。本発明の利点は、管理されるべきデータが優先順位付けされる（埋め込まれる）ことである。圧縮中または伝送中に十分なスペースまたは帯域幅を利用できないときには、重要度の低いデータを直ちに廃棄することができるため、合理的な損失性圧縮が可能である。

【０２５２】＜本発明の１パスシステム＞本発明は、システムの入力データを、それを受け取った時に完全に処理できる１パス変換を提供する。このようなシステムでは、データの処理が後続データに依存しない。画像を圧縮するために必要とされるメモリが、画像の長さに依存しない。本発明は、かかる依存性の排除により、データ全部の処理が完了する前に圧縮データを出力可能なシステムを提供する。

【０２５３】＜Ａ．１パス変換のためのデータ管理＞図３１は本発明の教えるところに従って帯状にラスタ順に圧縮中の画像の一部を示す。４レベルの分割を想定する。各ツリーは$2^4\times 2^4=16\times 16=256$個の係数を有する。しかし、本発明におけるウエーブレット変換の高域通過フィルタはオーバーラップしているので、各ツリーは２５６個より多数の入力画素に依存する。２タップ”１１”低域通過フィルタ（Ｌ）は全くオーバーラップを生じず、オーバーラップは全て６タップ”-1 -1 8 -8 1 1”高域通過フィルタ（Ｈ）に起因する。低域通過フィルタを３回、続いて高域通過フィルタを１回、直列的にかける場合（ＬＬＬＨ）に、最大のオーバーラップが生じる。低域通過フィルタを３回かける（ＬＬ

49

Ｌ）には、2^3＝８個の入力画素のサポートを必要とする。８×８画素サイズのサポート領域が図３１に示されている。高域通過フィルタが直列に入れられた時には、サポート領域は（６×2^3）×（６×2^3）＝４８×４８画素である。１つの４８×４８画素のサポート領域は、図３１に見られるように３６個の８×８ブロックからなる。

【０２５４】図３１に示した４８×４８画素サポート領域の係数が現在処理されているとする。このサポート領域の淡く陰を付けた部分は、前のサポート領域で既に利用された画素を表わす。サポート領域の外側の淡く陰を付けた部分は、前サポート領域で既に利用され、かつ、これから先のサポート領域で必要となる画素を示している。黒い１６×１６領域は、当該サポート領域のまだ利用されていない画素を含む部分である。同様に、濃く陰を付けた１６×１６領域は、まだ利用されておらず、次の４８×４８サポート領域で利用されることになる画素を含む。１個の３レベル１６×１６変換が計算され、他の８個の３レベル１６×１６変換についての前の結果がバッファから読み出され、そして変換の第４レベルがこれら９個の３レベル１６×１６変換に施される。この実行に要するバッファリングは、（２×画像幅＋３２）×１６画素に対する３レベル変換係数を格納するに足り、その上に１６ライン（１つの帯）バッファ分の画素を格納するに足りるということである。

【０２５５】図３２は１パスウエーブレットフィルタリングユニットの一実施例のブロック図であり、このユニットはフィルタ制御ユニット１３０１、メモリ１３０２及びフィルタ１３０３からなる。フィルタ１３０３は図２９に関連して説明したフィルタからなる。メモリ１３０２は図３１に関連して上に述べたメモリを指し、画素または係数を格納する。フィルタ制御ユニット１３０１は、メモリ１３０２とフィルタ１３０３の間のデータフローを決定する。フィルタ制御ユニット１３０１の動作は後述する。

【０２５６】図３３は別のウエーブレットフィルタユニットを示す。高速動作を達成するため、複数のフィルタを使用可能である。一実施例では、フィルタ１３０３は４個または５個の入力を必要とし（例えば、逆フィルタ、フォワードフィルタ）、２個の出力を出すので、必要とされるメモリ帯域幅はかなりになろう。メモリは、１ロケーションあたり複数の画素／係数を、また複数のバンク及び／または複数のポートを持つかもしれない。メモリインターフェイスユニット１４０１は、処理中に利用されるローカルデータ用の小バッファを提供することによって、必要とされるメモリ帯域幅を減らす。メモリインターフェイスユニット１４０１はまた、メモリ１３０２の入出力（Ｉ／Ｏ）とフィルタのＩ／Ｏとの間のマルチプレクス／デマルチプレクスを行なう。

【０２５７】フィルタリングのために必要とされるメモリ帯域幅に加え、メモリ１３０２への画素入力及びコンテキストモデルへの係数の出力のために余分な帯域幅が必要となるかもしれない。画素がラスタ順に入力されるならば、帯バッファのために余分なメモリが必要になろう。

50

【０２５８】メモリが１ロケーションあたり複数の要素（画素または係数）を格納する場合、水平方向または垂直方向に隣接する要素を１行または１列に格納するのではなく、１個のＮ×Ｎブロック（Ｎは２のべき乗）の要素が同じロケーションを共有すれば、必要なメモリアクセス及びバッファの量が減少しよい。こうすると、垂直方向アクセス及び水平方向アクセスのいずれにも等しく便利である。

【０２５９】図３４に示されるように、複数バンクのメモリも、それを水平方向アクセスと垂直方向アクセスで均等に利用できるように構成し得る。２バンクの場合、どちらかのバンクを選択するために用意される１ビットのバンク選択ビットが、例えば、垂直座標と水平座標の最下位ビットを論理和することによって作られるかもしれない。４バンクの場合、２ビットのバンク選択ビットが、水平座標と垂直座標の最下位２ビットを加算（２ビット加算器を用いてモジュロ４で）することによって作られるかもしれない。

【０２６０】図３５は、フィルタ制御ユニット１３０１（図３２）による２レベル分割を実現するための１パスフィルタ動作を示す。なお、説明の都合上、まず２レベル分割について論じてから、本発明の一般的手法を説明する。他の実施例では、３レベル、４レベル、またはそれ以上のレベル数の分割が用いられる。２レベル分割は、１ツリーあたり１６個の係数を持ち、前に利用されなかった１６個の入力画素について計算を必要とする。１ツリーの係数に対するフィルタリングは、入出力レートと調和すべく１６時間単位以内で成し遂げられる。この例では、必要とされるスループットである１単位時間あたり２フィルタリング動作を達成するため、並列動作する２個のフィルタが用いられる。図３５は、フィルタの前エッジが適用される各空間位置について、各フィルタリング動作が実行される時刻を示す数字を表わしている。

【０２６１】フィルタリングの順序はフィルタの前エッジで決まるので、フィルタリングは、あるツリーの係数の全部を生成しないうちに次のツリーの係数のどれかを生成する。ツリーの子のフィルタリングは親のフィルタリングの前に行なわれ、低域通過フィルタリングは対応の高域通過フィルタリングの前に行なわれる。フィルタリングは、Ａグループの係数に作用する。

【０２６２】レベル１の水平フィルタリングは時刻０から時刻７の期間に実行され、その結果は一時バッファに格納される。（各空間位置から２個の係数が得られる。）時刻２から時刻９の期間に、（第２のフィルタに

51

より）垂直フィルタリングが、バッファ内のデータ及びメモリから与えられる前水平フィルタリングによるデータに対して行なわれる（各空間位置毎に２回）。垂直フィルタリングは、２番目の水平フィルタリングが完了したら直ぐに開始できる。ＨＨ，ＨＬ，ＬＨ係数は（適当な時刻に）コンテキストモデルへ出力できる準備ができている。ＬＬ係数は次のレベルで使われる。

【０２６３】フィルタが２個だけでは、レベル１の水平フィルタリングが完了して１つのフィルタが利用できるようになる時刻８まで、レベル０の水平フィルタリングは開始できない。レベル０の垂直フィルタリングが完了し必要データ全部が得られてから１サイクル後の時刻１０まで、レベル０の水平フィルタリングは終了できない。次の時刻１１と時刻１２の間、レベル１の垂直フィルタリングを行なうことができる。

【０２６４】下記表１は、各時間単位における各フィルタの動作をまとめて示す。各エントリーの形式は、レベル番号、水平または垂直（”Ｈ”または”Ｖ”）、そして前エッジの空間位置である。垂直フィルタリング操作の入力は、添字”Ｌ”または”Ｈ”を用いて低域、高域が区別されている。なお、両方のフィルタは同じものであるから、一方のフィルタを水平フィルタリングに、他方のフィルタを垂直フィルタリングに、割り当てる必要はない。

【０２６５】

【表１】

表　1

Time	Filter 1	Filter 2
0	1H(0,0)	(idle)
1	1H(0,1)	
2	1H(2,0)	$1V_L(0,0)$
3	1H(2,1)	$1V_H(0,0)$
4	1H(0,2)	$1V_L(0,2)$
5	1H(0,3)	$1V_H(2,0)$
6	1H(2,2)	$1V_L(0,2)$
7	1H(2,3)	$1V_H(0,2)$
8	0H(0,0)	$1V_L(2,2)$
9	(idle)	$1V_H(2,2)$
10	0H(0,1)	(Idle)
11	(idle)	$0V_L(0,0)$
12		$0V_H(0,0)$
13		(idle)
14		
15		

52

【０２６６】レベル１水平フィルタリングは、次の入力画素群に対して時刻１１に再び開始することができるけれども、そうするとフィルタを入出力レートより高速に動作させることになる。本発明では、そのようにしないで、フィルタは遊び（アイドル）になり、そして次の入力画素群は時刻１６に始まることになろう。遊びのフィルタリングサイクルはメモリ転送のために利用される。遊びサイクルは、必要ならば、各入力画素群に対するフィルタリングの終わりに発生させるのではなくて、フィルタリングサイクル中に分散させてもよい。

【０２６７】２レベルの場合を考慮な入れて３レベルの場合を表２に示す。情報を１ページに表わして読み易くするため、２または４時間単位の連鎖が使われている。

【０２６８】

【表２】

表 2

Time	Filter 1	Filter 2
0-3	2H(0,0), 2H(0,1), 2H(2,0) 2H(2,1)	(idle),(idle) $2V_L(0,0)$, $2V_H(0,0)$
4-7	2H(4,0), 2H(4,1), 2H(6,0) 2H(6,1)	$2V_L(2,0)$, $2V_H(2,0)$, $2V_L(4,0)$, $2V_H(4,0)$
8-11	2H(0,2), 2H(0,3), 2H(2,2) 2H(2,3)	$2V_L(6,0)$, $2V_H(6,0)$, $2V_L(0,2)$, $2V_H(0,2)$
12-15	2H(4,2), 2H(4,3), 2H(6,2) 2H(6,3)	$2V_L(2,2)$, $2V_H(2,2)$, $2V_L(4,2)$, $2V_H(4,2)$
16-19	2H(0,4), 2H(0,5), 2H(2,4) 2H(2,5)	$2V_L(6,2)$, $2V_H(6,2)$, $2V_L(0,4)$, $2V_H(0,4)$
20-23	2H(4,4), 2H(4,5), 2H(6,4) 2H(6,5)	$2V_L(2,4)$, $2V_H(2,4)$, $2V_L(4,4)$, $2V_H(4,4)$
24-27	2H(0,6), 2H(0,7), 2H(2,6) 2H(2,7)	$2V_L(6,4)$, $2V_H(6,4)$, $2V_L(0,6)$, $2V_H(0,6)$
28-31	2H(4,6), 2H(4,7), 2H(6,6) 2H(6,7)	$2V_L(2,6)$, $2V_H(2,6)$, $2V_L(4,6)$, $2V_H(4,6)$
32-35	1H(0,0), 1H(0,1), 1H(2,0) 1H(2,1)	$2V_L(6,6)$, $2V_H(6,6)$, $1V_L(0,0)$, $1V_H(0,0)$
36-39	1H(0,2), 1H(0,3), 1H(2,2) 1H(2,3)	$1V_L(2,0)$, $1V_H(2,0)$, $1V_L(0,2)$, $1V_H(0,2)$
40-43	0H(0,0, (idle), 0H(0,1), (idle)	$1V_L(2,2)$, $1V_H(2,2)$, (idle), $0V_L(0,0)$,
44-47	(idle)	$2V_H(0,0)$, (idle), (idle), (idle)
48-51		(idle)
52-55		
56-59		
60-63		

【０２６９】表３は４レベルの場合を示す。１グループの係数につき２５６時間ユニットあるので、単純にするためフィルタリングのレベルと方向だけが示されている。

【０２７０】

【表３】

表 3

Time	Filter 1	Filter 2
0-1	Level 3 Horizontal	(idle)
2-127	Level 3 Horizontal	Level 3 Vertical
128-129	Level 2 Horizontal	Level 3 Vertical
130-159	Level 2 Horizontal	Level 2 Vertical
160-161	Level 1 Horizontal	Level 2 Vertical
162-167	Level 1 Horizontal	Level 1 Vertical
168	Level 0 Horizontal	Level 1 Vertical
169	(idle)	Level 1 Vertical
170	Level 0 Horizontal	(idle)
171	(idle)	Level 0 Vertical
172	(idle)	Level 0 Vertical
173-255	(idle)	(idle)

【０２７１】本発明のフィルタリング及びメモリサブシステムの出力は係数の系列であり、これは本発明のビット・シグニフィカンス埋め込み符号化を受ける。

【０２７２】＜Ｂ．１パスシステム用コンテキストモデル＞本発明の一実施例である１パスシステム用ビット・シグニフィカンス埋め込みコンテキストモデルでは、各ツリーが４つの部分に分けて処理される。ツリーのルートである最高レベルのＬＬ係数は、１パス水平順序符号化によって符号化される。ルートの３個の子から始まる３個のサブツリー（subtree）である最高レベルのＨ，ＨＬ，ＬＨ係数は、１パス統合空間／周波数モデリング及び１パス周波数ベースモデリングの両方によって処理される。ビット・シグニフィカンス埋め込みコンテキストモデルがデータ全部を処理するより前に符号化データが出力されるように、係数は符号化される。

【０２７３】＜１パス・シグニフィカンス（significance）ツリー＞ゼロツリーコンテキストモデルは１パスシステムに利用できない。ゼロツリーは全ての係数を保持する１つのリスト（または複数のリスト）を必要とし、またゼロツリーはリストに複数のパスを作る。もう一つ

55

の周波数ベースモデルである１パス・シグニフィカンスツリーは、全係数を保持するリストを全く必要としない。１パス・シグニフィカンスツリーとゼロツリーとの間のもう一つの相違点は、シグニフィカンスツリーは全ての子を処理した後に、その親を処理しつつデシジョンを生成するのに対し、ゼロツリーは親を先に処理するということである。本発明のコンテキストモデルが図３６にブロック図の形で示されている。このコンテキストモデル１７００は、符号／絶対値ユニット１０９（図３７）とシグニフィカンス（significance）ユニット１７０２という２つの処理ユニットを含む。コンテキストモデル１７００はまた、絶対値メモリ１７０１とツリー（tree）メモリ１７０３という２つのメモリを（メモリ制御論理も）用いる。これら２つのメモリユニットのそれぞれは、複数の記憶エリアによって高速動作中に交互に使用できるように構成してもよい（すなわち、一方がデータ書き込み中である時に、他方は読み出し中であるかまたは空である）。

【０２７４】絶対値メモリ１７０１は、ツリー中の係数をシグニフィカンスに基づいた順序に、例えば、係数の絶対値に基づいた順に並べ替える。この並べ替えは、各可能な絶対値のためのキュー（queue）を作ることによって達成される。シグニフィカンスユニット１７０２は、係数をシグニフイカンス（例えば絶対値）の順に受け取り、Ａ－パスアルゴリズムを処理する符号化器のためのデシジョンを生成する。ツリーメモリ１７０３はシグニフィカンスユニット１７０２に接続され、オールゼロの後のゼロツリーを削除する。

【０２７５】以下の説明は、係数が１８ビットであり、かつ入力データが４レベル分割されていると仮定している。

【０２７６】符号－絶対値フォーマッティングユニット１０９の一実施例が図３７に示されており、これは入力

56

した係数を符号－絶対値形式へ変換する。符号－絶対値フォーマッティングユニット１０９は、１８ビットの係数を受け取るように接続されており、インバータ１８０１、マルチプレクサ（ＭＵＸ）１８０２、プライオリティ（priority）エンコーダ１８０３、及びカウンタ１８０４からなる。符号－絶対値ユニット１０９は、シグニフィカンス（significance）表示（例えば５ビット値）、入力係数の仮数（例えば１７ビット）、入力係数の符号（sign）（例えば１ビット）、及びカウンタ１８０４からのインデックス（例えば７ビット）を出力する。

【０２７７】ＭＵＸ１８０２は、符号－絶対値フォーマッティングユニット１０９に直接入力した係数の１７ビットと、この１７ビットをインバータ（２の補数器）１８０１により反転したものを受け取るように接続されている。ＭＵＸ１８０２の選択入力に入る符号ビット（係数のビット１７）に基づき、２つの入力のうちの正のものが仮数として出力される。

【０２７８】符号－絶対値フォーマッティングユニット１０９はプライオリティエンコーダ１８０３を使い、各係数の最初の有効ビットを決定する。各係数の最初の有効ビットに基づいて、１つのシグニフィカンスレベルを係数に関係付けることができる。

【０２７９】カウンタ１８０４は、１つのインデックスを現在のツリー要素に関係付けするために利用される。４レベル分割の場合、そのインデックスは０から８４まで変化する（１つのサブツリーの要素数は１＋４＋１６＋６４＝８５であるから）。入力係数はツリー順であり、この例では、最初に親、最後が子の順序であると仮定している。異なった分割レベルによる係数は、表４に示す通りである。

【０２８０】

【表４】

表　4

Level	Coefficients index
0	0
1	1, 22, 43, 64
2	2, 7, 12, 17, 23, 28, 33, 38, 44, 49, 54, 59, 65, 70, 75, 80
3	3...6, 8...11, 13...16, 18...21, 24...27, 29...32, 34...37, 39...42, 45...48, 50...53, 55...58, 60...63, 66...69, 71...74, 76...79, 81...84

【０２８１】図３８は、絶対値メモリ１７０１の一実施例のブロック図である。１つのカウンタ（１９００～１９１６）とメモリ（１９２０～１９３６）が、可能な各シグニフィカンスレベルに関係付けられている（ただし、符号化される必要のないゼロ係数に対しては何も必要でない）。例えば、カウンタ１９１６とメモリ１９３６はシグニフイカンス（significane）レベル１７に関係付けられている。一実施例では１６のシグニフィカンスレベルがある。したがって、１７個のカウンタ１９００～１９１６と、それに関連した１７個のメモリ１９２０～１９３６がある。

【０２８２】一実施例では、各メモリは、サブツリーの可能な各係数につき、８５のロケーションを持たなければならない（各サブツリーは８５個の係数を含むから）、しかし、そのメモリサイズは便宜上、２のべき乗、例えば１２８に丸められるであろう。メモリの各エ

ントリーは１ビットの符号ビット、７ビットのインデックスビット及びＮビットの絶対値ビットを格納できる（Ｎはシグニフィカンスレベルである）。固定幅のメモリを使いたいときには、シグニフィカンスレベル１６と０のエントリー、シグニフィカンスレベル１５と１のエントリー、等々を結合することができ、そうすることにより、各ワードは合計３２ビットの２つのエントリーを含む。当然のことながら、シグニフィカンスレベル数が奇数の場合、あるワードは１つのエントリー（この例ではレベル７）しか含まない。

【０２８３】符号－絶対値フォーマッティングユニット１０９より受け取った符号（sign）、インデックス及び仮数の値は、適切なメモリの、それに関係付けられたメモリカウンタにより与えられるアドレスに書き込まれる。この関係付けられたカウンタは、該当シグニフィカンスレベルの次の係数が次ロケーションに格納されるようインクリメントされる。

【０２８４】記憶内容は、シグニフィカンスの降順にメモリ１９０２～１９２６のそれぞれより読み出される。各係数の出力は、その仮数、符号（sign）及びインデックス出力からなる。最高のシグニフィカンスレベル（例えばレベル１６）のカウンタが０でなければ、このカウンタはデクリメントされ、メモリはそのアドレスが読み出される。これはカウンタ値が０になるまで繰り返される。そして、次のシグニフィカンスレベル（例えばレベル１５）が検討される。全てのカウンタが０までデクリメントされ全部のメモリが空になるまで、各シグニフィカンスレベルが順に検討される。

【０２８５】リアルタイムのシステムにおいては、２バンクのカウンタ及びメモリを用いるのが望ましいかもしれない。そうすれば、一方のバンクを、他方のバンクが出力のために使われている時に入力のために用いることができる。

【０２８６】カウンタは、ＬＩＦＯ（後入れ先出し）が実現されるように関連したメモリをアドレスする。ＬＩＦＯは、サブツリーが親から先に入力される場合には適する順序である。そうではなく、サブツリーが子から先に入力されるのであれば、カウンタの動作はＦＩＦＯ（先入れ先出し）を実現するように変更されることになろう。

【０２８７】図３９はシグニフイカンスユニット１７０２の一実施例のブロック図である。図３９において、インデックスカウンタ２００１はサブツリーの各係数を、子から先に順に進めるために用いられる。一実施例では、インデックスカウンタ２００１は８４に初期化されてから０までカウントダウンする。シグニフィカンス（significance）カウンタ２００４は、最大のシグニフィカンスレベル（この例では１６）から始まって、インデックスカウンタが１サイクルする（８４に戻る）毎にカウントダウンすることにより、ビットプレーンを管理

する。ある特定のインデックスのレベルが、前記表４に示した機能を遂行するインデックスーレベル論理２００３によって決定される。

【０２８８】絶対値メモリ１７０１は、シグニフィカンスカウンタ２００４の出力によりイネーブルされたメモリ内にある次の係数のインデックス、絶対値及び符号を提供する。メモリから入力したインデックスがインデックスカウンタ２００１のインデックス出力と同一であると、一致論理２００２が非ゼロ出力表示をアサートする。この非ゼロ出力表示は、次サイクルで絶対値メモリが次のインデックス等を供給すべきことを意味する。一致がとれないときには、不一致表示がデシジョン（decision）ジェネレータ２００８へ送られる。

【０２８９】一実施例では、フラグ０，フラグ１，フラグ２として示した３個のフリップフロップ２００５，２００６，２００７が、非ゼロデータの管理のために用いられ、また、これらフリップフロップは分割レベル０，１，２にそれぞれ割り当てられている。なお、必要なフリップフロップの個数は、分割レベル数より１つ少ない。フリップフロップ２００５～２００７は、最初にクリアされる。一致論理２００２から出る非ゼロ信号がアサートされた時に、フリップフロップ２００５～２００７中の現在のレベルより小さいレベルに割り当てられたフリップフロップがセットされる。現在のレベルに割り当てられたフリップフロップはクリアされる。レベルはインデックスーレベル論理２００３によって与えられるが、このインデックスーレベル論理２００３はインデックスカウンタ２００１によって与えられたインデックスに応じてレベルを提供する。

【０２９０】”符号化済み”フラグ（ある実施例ではレジスタファイル）が各インデックス毎に１ビットずつ記憶される。非ゼロ信号がアサートされた時に、符号化済みフラグ記憶内の現在インデックスカウンタ値に関連したビットがセットされる。そうでなくて、シグニフィカンスカウンタ値が最大値になれば、関連したビットはクリアされる。そうでなければ、そのビットの値はそのままの値を維持する。符号化済みフラグ記憶から出る符号化済み出力信号は、現在のインデックスに関連したビットの新値と同じである。なお、他の実施例では、符号化済みフラグは用いられず、したがって符号化済み信号も全く用いられない。

【０２９１】一実施例では、デシジョンジェネレータ２００８は、現在レベルが３であり、かつ前レベルが３でなかったか判定する。この判定に応じて、デシジョンジェネレータ２００８は開始出力をアサートし、開始レベル出力は前レベルである。非ゼロ信号がアサートされたならば、デシジョンジェネレータ２００８は”有意”なるデシジョンを出力し、また、符号（００、０１）と仮数を出力する。そうでなくて、符号化済み入力がアサートされたならば、デシジョンは出力されない。そうでな

59

くて、現在レベルに割り当てられたフラグフリップフロップがセットされたならば、デシジョンジェネレータ２００８は”無意、有意な子あり”なるデシジョン（１０）を出力する。そうでなければ、デシジョンジェネレータ２００８は、”無意、かつ子も無意”なるデシジョン（１１）を出力し、かつオールゼロ信号をアサートする。

【０２９２】なお、周波数ベースモデリングと水平１パス統合空間－周波数モデリングの両方を実現するためには、シグニフィカンスユニット１７０２に対し以下の変更がなされる。シグニフィカンスカウンタ２００４はある閾値と比較され、カウンタ値が同閾値を超えるときにのみオールゼロ信号がアサートされる。

【０２９３】一実施例では、ツリーメモリ１７０３（図４０、後述）に対するシグニフィカンスカテゴリー入力は、シグニフィカンスカウンタ２００４の出力である。この実施例のコンテキストモデル（すなわちビット・シグニフィカンス埋め込みユニット）では、シグニフィカンスカテゴリーはビットプレーン数に基づいており、１７の異なったシグニフィカンスカテゴリーがある。これは任意に選んでよい。別の実施例では、シグニフイカンスカテゴリーを減らすためビットプレーンが結合されるかもしれない。また、より多くのシグニフィカンスカテゴリーを作るために、レベル情報をビットプレーン情報に追加することもできる。シグニフィカンスカテゴリーを増やすと、より優れた損失性圧縮が可能になるであろう。他方、シグニフィカンスカテゴリーが少ないほうが、ハードウエアの複雑さが軽減されるであろう。

【０２９４】図４０は本発明のツリーメモリユニットの一実施例のブロック図である。図４０において、メモリ２１０１は、デシジョンと、可能な各デシジョンのためのシグニフィカンス表示を記憶するための適切なスペースを有する。一実施例では、４レベル分割、７シグニフイカンスレベルの場合で、メモリ２１０１のロケーション数は８５×１７＝１４５５である。

【０２９５】メモリ２１０１をアクセスするためアドレスが生成される。カウンタ２１０２は最初は０である。デシジョンジェネレータ２００８がオールゼロ入力をアサートしない時に、カウンタ２１０２の値がメモリのアドレスのために用いられる。デシジョンジェネレータ２００８が開始入力をアサートすると、カウンタ２１０２の現在値は開始レベルに応じてレジスタ２１１０～２１１２中の１つのレジスタに格納される。開始レベルは一種の選択機構として作用する。そして、カウンタ２１０２はインクリメントされる。

【０２９６】デシジョンジェネレータ２００８がオールゼロ入力をアサートした時に、レベル入力により選択されたレジスタ（すなわち２１１０、２１１１、２１１２）中の値がメモリ２１０１のアドレスのために用いられ、この値に１を加えた値がカウンタ２１０２にロードされる。これは、無意の親の無意の子のために使用されたメモリロケーションを無視させる。

60

【０２９７】メモリ出力期間に、出力すべきロケーションのアドレスを提供するためカウンタ２１０２はデクリメントされる。出力（及びデクリメント）はカウンタ２１０２が０になった時に止まる。メモリ２１０１の出力はエントロピー符号化器に受け取られ、エントロピー符号化器は指定されたシグニフィカンスでデシジョンを適切に符号化する。

【０２９８】リアルタイム動作のためには、２個のツリーメモリユニットを用い、一方を入力に利用し、同時に他方を出力に利用するようにすることができる。

【０２９９】＜係数アラインメント（alignment）＞本発明の一実施例においては、ゼロツリーコンテキストモデルは非正規化$1+Z^{-1}$低域通過フィルタを使用する。しかし、ゼロツリーコンテキストモデルは、例えば次に示したような正規化フィルタと一緒に使用し得る。

【０３００】

【数３６】

$$\frac{1+Z^{-1}}{\sqrt{2}}$$

【０３０１】正規化フィルタを使用するためには、図４１のフォワード・ウエーブレットフィルタ１０００と（周波数ベース）コンテキストモデル１０５の間のアラインメント（alignment）ユニット２２００のようなアラインメントユニットを用いて、非正規化フィルタにより獲得したエネルギー（あるいは失ったエネルギー）を補償し、圧縮を改善することができる。アラインメントは損失性動作のための非一様量子化を許容するので、アラインメントは損失性画像復元の視覚的品質を改善できる。一次元の場合、ツリーの各レベルからの係数のアラインメントは様々であろう（除数＝$2^{1/2}$，２，$2\times 2^{1/2}$，４、乗数＝$2\times 2^{1/2}$，２，$2^{1/2}$，１）。二次元の場合、除数は２，３，８，１６、乗数は８，４，２，１であろう。

【０３０２】アラインメントは単に同様の符号化用バイナリデシジョンをグループにするためであるので、厳密な正規化値を用いることは重要でない。復号期間にはアラインメントを逆にしなければならないので、乗算と除算の両方が必要になる。２のべき乗の乗数（因数）／除数を用いると、ハードウエアによる効率的なシフト操作を行なえるようになろう。係数に２のべき乗が掛け合わされる場合、シグニフィカンスの小さい追加された０ビットは符号化する必要がない。

【０３０３】しかし、アラインメント乗数／除数を２のべき乗に制限する代わりに、$2^{1/2}$＝～１．５または$2^{1/2}$＝～２÷１．５というような近似を、以下の方法とともに用いることができる。乗数／除数で係数を乗算／除算するのではなく、代わりに”重要な”係数だけが乗

61

数／除数によりスケール（scale）されてもよい。符号－絶対値フォーマッティングユニットは、図４２に示すように、（１）最も上位の”１”ビットの次位ビットも”１”のときに最も上位の”１”ビットの位置を返し、（２）そうでなければ最も上位の”１”ビットの一つ下の位置を返す”１．５”プラオリティエンコーダ２３０１を含むように変更し得る。３ビット入力の場合の”１．５”プライオリティエンコーダの真理値表を表５に示す。

【０３０４】

【表５】

表　５

Input (Binary)	Output
001	0
010	0
011	1
100	1
101	1
110	2
111	2

【０３０５】現在のインデックス値で指示される係数のレベルに応じて、マルチプレクサ２３０２は標準的なプライオリティエンコーダ１８０３または”１．５”プライオリティエンコーダ２３０１のどちらかより与えられるシグニフイカンス（significance）を選択する。”１．５”アラインメントが用いられる時には常に、仮数はＮ＋１ビットである（Ｎはシグニフィカンス値）。そうでない時には、仮数はＮビットである。

【０３０６】アラインメントユニット２２００は、シフタとして作用する２入力乗算器からなり、１または２によるアラインメントを実施できる。これと、符号－絶対値フォーマッティングユニットにより提供される”１．５”アラインメントとを組み合わせることで、１，１．５，または３のアラインメントが可能である。１，１．５または３は、単純であるので（すなわち２のべき乗である）、一次元信号のために必要な乗数のよい近似値である。（画像のような二次元信号の場合、それら数はより単純である。）　復号期間において、”１．５”プライオリティエンコーダが用いられる時には仮数の第（Ｎ＋２）ビット（符号されない）は第（Ｎ＋１）ビットの補数である。

【０３０７】係数アラインメントを、ゼロツリーのチューニング及びより精度の高い非一様量子化のために利用できる。画像（二次元信号）の場合、ＲＴＳ変換の一実施例は、図５６に示した数を周波数帯域に掛け合わせることによって、係数を調整する。これらの数を掛けると、ＲＴＳ変換が、その正確な復元ウエーブレットの極

62

めて精度のよい近似となる。

【０３０８】エントロピー符号化器は、アラインメントプロセスが効率的になるよう考慮しなければならない。

【０３０９】部分ビットプレーンによる周波数ベースコンテキストモデル

周波数ベースモデリングの別の方法は、部分ビットプレーンもしくは部分的な有意ビット（bits of significance）を利用する。これを実現する一つの方法は、各ビットプレーンを２回処理することであり、その結果、パスとしてＡ１－パス、Ｂ１－パス、Ａ０－パス、Ｂ０－パスがある。なお、これらパスの名称は、Ａ１－パスが”１１”で始まる係数を扱い、Ａ０－パスが”１０”で始まる係数を扱うことから選ばれた。

【０３１０】ビットプレーンＳに対するＡ１－パスの期間において、Ａ－グループの係数が有意であるのはビットＳ，Ｓ－１がともに非ゼロのときだけである。Ａ２－パスの期間において、Ａ－グループの係数はビットＳが非ゼロのきに有意である。最上位の２ビットは知れているので、Ｂ１－パスとＢ０－パスはＳ－１ビットを処理するだけでよい（Ｓ＝０は最下位ビットプレーンであると仮定する）。

【０３１１】１つおきの部分ビットプレーンは、因数が１．５または２／１．５と異なるから、異なるレベルに対するアラインメントは、各レベル毎に必要な部分ビットプレーンをグループにすることによって達成できる。

【０３１２】部分ビットプレーンは、周波数ベースコンテキストモデルに利用された親／子関係によってデータのより精密なモデリングをもたらす。さらに精密なモデリングのために、３つ以上のパス、例えば４つまたは８つのパスを用いてもよい。例えば、４パスの場合、Ａ１１－パスは”１１１”で始まる係数を扱うことになろう。そして他のパスは”１１０”，”１０１”，”１００”を扱うことになろう。より精度の悪いモデリングもまた利用されるかもしれない。例えば、１つおきのビットプレーンに対しのみ１つのパスが行なわれるかもしれない。このような低精度モデリングの場合には、より多くのビットがＢ－グループによって符号化される。

【０３１３】＜Ｃ．１パスシステムのための符号化器及びメモリ／チャネル管理＞データ全部をメモリに格納するシステム及びデータをチャネルで伝送するシステムに関して、１パスシステムにおける符号化データのメモリ管理を提案する。１パスシステムでは、符号化データは”埋め込みカジュアル（embedded casual）”アクセスすることができて、シグニフィカンスの高いデータを損なわずにシグニフィカンスの低いデータを廃棄できるよう格納されなければならない。符号化データは可変長であるので、動的メモリ割り当てを利用できる。

【０３１４】本発明の一実施例では、埋め込み符号化スキームは１８個のビットプレーンを用い、したがって１８個の重シグニフィカンスレベルをデータに割り当て

る。１パスシステムの符号化器は、”埋め込みカジュアル（embedded casual）”でなければならない。すなわち、あるビットプレーンに対応するイベントの復号に、それにより下位のビットプレーンの情報を必要としない。１パスの場合、普通、あるツリーのビット全部が符号化された後に次のツリーのビットが符号化されるから、シグニフィカントの異なるビットが分離されない。内部状態（internalstate）を用いるハフマン符号化器のような符号化器にとって、このことは問題ではない。しかし、多くの圧縮率の優れた高度な圧縮器は内部状態を用いる。

【０３１５】これら符号化器に関する当該問題を解決する一方法は、１８個の異なる符号化器、多分Ｑ－コーダチップ、を用いることである。９個のＱ－コーダチップを使用できるであろう手法が、”Data Compression for Recording on a RecordMediumなる発明の名称で１９９２年3月１７日発行された米国特許第５，０９７，２６１号（Langdon, Jr）に述べられている。より優れた方法は、単一の物理符号化器で様々な仮想符号化器を実現するためにパイプライン方式の符号器、例えば”Method and Appartus for Parallel Decoding and Encoding of Data”なる発明の名称で１９９３年1月１０日受理された米国特許出願第０８／０１６，０３５号に述べられている符号化器を用いる。このような符号化器では、各確率毎の複数のビットジェネレータ状態がデータの一部に割り当てられる。例えば、１８ビットデータの場合、１８個の状態中の各状態がある１つの特定のビットプレーンに割り当てられることになろう。符号化器内部のレジスタもまたデータの各部分に割り当てられる。この符号化器においては、インターリービングは行なわれない。すなわち、データの各部分は単にビット詰めされる。

【０３１６】複数の物理符号化器または仮想符号化器を備えた実施例では、データの各部分にメモリが割り当てられる。圧縮が完了した時に、割り当てられたメモリとその内容を記述する連結リストが結果として得られる。

【０３１７】メモリがオーバーフローすると、メモリ割り当てルーチンはより重要な情報をそれより重要でないデータに上書きさせる。例えば、数値データの最下位ビットが初めに上書きされることになろう。メモリの割り当てられ方を記述する情報が、符号化データのほかに、格納されなければならない。

【０３１８】図４３は、３つのシグニフィカンスカテゴリーのための動的メモリ割り当てユニットの例を示す。本発明をいたずらに難解にしないため３カテゴリーしか述べないが、一般的には、８カテゴリー、１６カテゴリー、１８カテゴリーというように、もっと大きな数のカテゴリーが用いられるであろう。レジスタファイル（または他の記憶手段）４８１が、各シグニフィカンスカテゴリー毎のポインタ（現在ポインタ）のほかに、次の空きメモリロケーションを指示するための別のポインタ

（フリーポインタ）を保持する。メモリ４８２は一定サイズのページに分割される。

【０３１９】最初、１つのシグニフィカンスカテゴリーに割り当てられた各ポインタは、あるメモリページの先頭を指し示し、また、フリーポインタは次に利用できるメモリページを指し示す。シグニフィカンスカテゴリーで区別された符号化データは、対応したポインタによりアドレスされたメモリロケーションに格納される。そして、このポインタは、次のメモリロケーションを指すようにインクリメントされる。

【０３２０】ポインタが現在ページの最大値に達した時に、フリーポインタに格納されている次の空きページの先頭のアドレスが、リンクとして現在ページと一緒に格納される。一実施例では、この目的のために、符号化データのメモリまたは独立したメモリもしくはレジスタファイルが用いられるかもしれない。次に、現在ポインタは次のフリーページを指すように設定される。フリーポインタはインクリメントされる。これらのステップにより、１つの新しいメモリページがある特定のシグニフィカンスカテゴリーに割り当てられ、かつ、ある共通のシグニフィカンスカテゴリーのためのデータを含むメモリページのリンクを得られる結果、復号期間に割り当て順序を確認することができる。

【０３２１】メモリの全ページが使用中であり、しかも、メモリ内のシグニフィカンスが最小のデータよりシグニフィカンスが大きいデータがさらに存在する時には、メモリの再割り当てが行なわれるかもしれない。そのような再割り当てのための３つの手法について述べる。いずれの手法にあっても、最小のシグニフィカンスのデータに割り当てられているメモリが、それより高いシグニフィカンスのデータへ再割り当てされ、もはや最小シグニフィカンスのデータは格納されない。

【０３２２】第１の手法は、最小シグニフィカンスのデータに現在使われているページが単純にシグニフィカンスの高いデータに割り当てられるだけである。最も一般的なエントロピー符号化器は内部状態情報を利用するので、そのページに前に格納されていた最小シグニフィカンスのデータは全部失われる。

【０３２３】第２の手法は、最小シグニフィカンスのデータによって現在使われているページは、それより大きいシグニフィカンスのデータに割り当てられる。第１の手法と違い、ポインタは当該ページの末尾を指すように設定され、大きいシグニフィカンスのデータが当該ページに書き込まれるにしたがい対応ポインタはデクリメントされる。この手法は、大きなシグニフィカンスのデータがページ全体を必要としないときには、ページの先頭にある最小シグニフィカンスのデータが保存されるという利点を有する。

【０３２４】第３の手法は、最小シグニフィカンスのデータの現在ページを再割り当てするのではなく、最小シ

65

グニフィカンスのデータの任意のページを再割り当てすることができる。そのためには、全ページの符号化データが独立に符号化される必要があり、このことは圧縮率の低下を招くかもしれない。また、全てのページの先頭に対応する非符号化データが識別される必要もある。最小シグニフィカンスのデータの任意のページを捨てることができるので、より大きな量子化のフレキシビリティを得られる。

【０３２５】画像の複数領域にわたって一定の圧縮率を達成するシステムでは、上記第３の手法は特に魅力的であろう。ある指定した数のメモリページを画像の１つの領域に割り当てることができる。シグニフィカンスの小さなデータが保存されるか否かは、特定の領域において達成される圧縮率によって決まる。なお、ある領域に割り当てられたメモリは、非損失性圧縮が必要とするメモリがそのメモリ量より少ないならば、完全には利用されない。画像のある領域に対し一定の圧縮率を達成すれば、その画像領域へのランダムアクセスをサポートすることができる。

【０３２６】圧縮が完了した時に、そのデータを必要ならばシグニフィカンス順にチャネルまたは記憶装置へ送ってよい。そうすれば、様々なリンク（link）とポインタはもう必要ではなく、また、マルチパス復号を行なってもよい。あるいは、１パス復号のために、各シグニフィカンス毎にデータへのポインタを保存することができる。

【０３２７】用途によっては、いくつかのシグニフィカンスカテゴリーは使われないかもしれない。例えば、１２ビットの医用画像に対して１６ビットの圧縮器が使用されるかもしれないが、その場合、ビットプレーン１５... １２に対応したシグニフィカンスカテゴリーは使用されないであろう。ページが大きく、かつ、多くのシグニフィカンスカテゴリーが使用されない例では、（いくつかのカテゴリーが使われないことをシステムが予め知らない時には）それらの使用しないカテゴリーにメモリを割り当てる必要はないのであるから、メモリを浪費することになろう。このメモリ浪費に対するもう一つの解決策は、各シグニフィカンスカテゴリー毎のカウントを保持するための小メモリ（もしくはレジスタ）を使用することであろう。このカウントは、他のデシジョンが発生する前に発生した”無意，有意な子なし”デシジョンの数を記録するであろう。これらのカウンタの格納のため必要となるメモリは、不使用のシグニフィカンスカテゴリーに利用されるメモリと”相殺”されなければならない。

【０３２８】システムにおいて利用可能なメモリ全量をより上手に利用するため、各ページにその両端からデータを書き込む機能を利用することができる。全ページが割り当てられている時に、一端側に十分な空きスペースのあるページを、その端から利用するように割り当てる

66

ことができる。ページの両端を利用する機能は、２種類のデータがぶつかるロケーションの管理コストと比較衡量されなければならない。もっとも、一方のデータ種類が重要でなく単純に上書きされてよい場合は別である。

【０３２９】＜チャネルの利用＞データがメモリに格納される代わりにチャネルで伝送され、かつ、固定サイズのメモリページが利用される（１つのシグニフィカンスカテゴリーにつき１ページだけ必要とされる）システムにおいて、あるメモリページが一杯になった時に、そのページはチャネルで伝送され、そして、伝送されるとすぐにメモリロケーションは再び利用できる。用途によっては、メモリのページサイズはチャネルで用いられるデータパケットのサイズまたはパケットサイズの倍数とすることができる。（なお、一実施例では、１シグニフィカンスレベルあたり２ページを使用できるので、一方のページにデータを書き込みながら他方のページをチャネルへ出力するため読み出すことができる。）

ある通信方式では、例えばＡＴＭ（非同期転送モード）では、パケットにプライオリティを割り当てることができる。ＡＴＭは、プライマリとセカンダリの２つのプライオリティレベルを有する。セカンダリパケットは十分な帯域幅を利用できるときだけ伝送される。どのシグニフィカンスカテゴリーがプライマリであるか、どのシグニフィカンスカテゴリーがセカンダリであるかを判断するために、閾値を利用することができる。もう一つの方法は、ある閾値よりシグニフィカンスの小さいシグニフィカンスカテゴリーを伝送しないように、符号化器において閾値を利用することであろう。

【０３３０】＜ピーク誤差を制限した損失性圧縮＞ある種の用途では、完全な（非損失の）復元は必要でない。指定された最大のピーク誤差で圧縮を成し遂げることが望ましいかもしれない。ピーク誤差を±Ｅと仮定する。これは、圧縮データを切り捨て、所望の精度を得るのに必要でないシグニフィカンスの小さなデータ全てを廃棄することによって達成できる。

【０３３１】指定された最大ピーク誤差の圧縮を遂行するもう一つの方法は、２×Ｅ＋１以下の値で、圧縮すべき画像の各画素を除算（整数除算）することである。復元中に、画像の各画素は

出力画素＝（２×Ｅ＋１）×入力画素＋Ｅ

と処理される。（あるいは、伸長中にＥを加算する代わりに、圧縮中に２×Ｅ＋１で割算する前に減算が行なってもよい。）

指定された最大ピーク誤差の圧縮を達成するもう一つの方法は、除算と乗算をシフトで置き換えることである。そのシフト量は、

【０３３２】

【数３７】

$$\lfloor \log 2(2xE+1) \rfloor$$

67

【０３３３】である。シフト操作は便利であるので、（ピーク誤差に代わる）より好ましい誤差仕様は（－２^{n}<エラー≦２^{n}］なる形の誤差であろう。

【０３３４】上に述べたことは、損失性画像圧縮の技術分野において周知である係数の量子化と混同すべきでない。多くの損失性圧縮システム（例えばＪＰＥＧ）においては、変換領域係数は最大ピーク誤差を割り付けられるが、これは画像のピーク誤差を間接的に規制するに過ぎない。決定的な違いは、本発明が画素に対して量子化を行ない、かつ、係数の非損失性圧縮を使うことである。

【０３３５】変換領域量子化も利用することができる。多くの係数は、変換の複数レベルにわたるピーク誤差の伝搬に対して影響がある。子のない高域通過係数に関するピーク誤差への影響を決定することは容易である。

【０３３６】一次元信号を最大ピーク誤差±Ｅで符号化することを考える。これは、最も精細な高域通過係数を±２Ｅに量子化することで達成できる。二次元信号の対しては、高域通過フィルタが２回かけられるので、最も精細なＨＨ係数を±４Ｅに量子化すればよい。

【０３３７】入力画像の量子化による方法の一つの代案は、エントロピー符号化器に対するデシジョンを規制することである。一例は次のとおりである。各係数について、係数を０に設定しても、当該係数に影響されるいかなる画素にも最大誤差を超える誤差が生じないならば、当該係数は０に設定される。ある具体例では、特定の係数のみ、多分、子を持たないＡＣ係数だけがテストされる。係数は、一度に一つしか検討しない贅沢なやり方で検討される。ほかのやり方は、係数の小グループを考え、グループの可能な最大の部分集合を０にするようにする。

【０３３８】

【表６】

表6　マスク

Bitplane	Mask (binary)
7	00000000
6	10000000
5	11000000
4	11100000
3	11110000
2	11111000
1	11111100
0	11111110

【０３３９】

68

【発明の効果】以上、実施例に関連して詳細に述べたところから明らかなように、本発明によれば、良好なエネルギー集中を得られる変換を用いた効率的な符号化装置と、それに関連した効率的な圧縮を可能にするウエーブレット変換フィルタとを実現することができ、したがって、効率的な非損失性／損失性データ圧縮／伸長システムを提供できる等の効果を得られる。

【図面の簡単な説明】

【図１】本発明の符号化システムの符号化部の一実施例を示すブロック図である。

【図２】本発明のビット・シグニフィカンス埋め込みブロックの一実施例を示すブロック図である。

【図３】ウエーブレット分析／合成システムのブロック図である。

【図４】非オーバーラップ最小長可逆フィルタを用いてフィルタリングする変換システムのフォワード変換とリバース変換の説明図である。

【図５】レベル１分割の結果の説明図である。

【図６】レベル２分割の結果の説明図である。

【図７】レベル３分割の結果の説明図である。

【図８】レベル４分割の結果の説明図である。

【図９】３レベル・ピラミッド変換の一例を示すブロック図である。

【図１０】２次元・２レベル変換の一例を示すブロック図である。

【図１１】２次元・２レベル変換の別の例を示すブロック図である。

【図１２】本発明の可逆ウエーブレットを用いる圧縮・伸長システムを示すブロック図である。

【図１３】本発明の可逆ウエーブレットを用いる強調及び分析システムを示すブロック図である。

【図１４】ウエーブレット係数のツリー構造を示す図である。

【図１５】本発明の単一リスト・ゼロツリー符号化プロセスの一実施例を示すフローチャートである。

【図１６】図１５に示したフローチャートの続きを示すフローチャートである。

【図１７】本発明の単一リスト・ゼロツリー符号化プロセスの他の実施例を示すフローチャートである。

【図１８】図１７に示したフローチャートの続きを示すフローチャートである。

【図１９】本発明のゼロツリー水平復号プロセスのためのＡ－パスの一実施例を示すフローチャートである。

【図２０】本発明のゼロツリー水平復号プロセスのためのＡ－パスの他の実施例を示すフローチャートである。

【図２１】本発明の単一リスト水平符号化プロセスのためのＡ－パスの一実施例を示すフローチャートである。

【図２２】本発明の単一リスト水平符号化プロセスのためのＡ－パスの他の実施例を示すフローチャートである。

69

【図２３】本発明の単一リスト水平復号プロセスのためのＡ－パスの一実施例を示すフローチャートである。
【図２４】本発明の単一リスト水平復号プロセスのためのＡ－パスの他の実施例を示すフローチャートである。
【図２５】本発明のゼロツリー水平符号化及び単一リスト水平符号化のプロセスのためのＢ－パスの一実施例を示すフローチャートである。
【図２６】本発明のゼロツリー水平符号化及び単一リスト水平符号化のプロセスのためのＢ－パスの他の実施例を示すフローチャートである。
【図２７】本発明のゼロツリー水平復号及び単一リスト水平復号のプロセスのためのＢ－パスの一実施例を示すフローチャートである。
【図２８】本発明のゼロツリー水平復号及び単一リスト水平復号のプロセスのためのＢ－パスの他の実施例を示すフローチャートである。
【図２９】本発明のフォワード・ウエーブレットフィルタの一実施例を示すブロック図である。
【図３０】本発明のリバース・ウエーブレットフィルタの一実施例を示すブロック図である。
【図３１】４レベル・ピラミッド分割のためのラインバッファ内の画像及び係数を説明するための図である。
【図３２】フィルタ制御ユニットを使用する本発明のウエーブレットフィルタユニットの一実施例を示すブロック図である。
【図３３】フィルタ制御ユニットを使用する本発明のウエーブレットフィルタユニットの他の実施例を示すブロック図である。
【図３４】水平、垂直アクセスをサポートするためのメモリバンクの割り当てを２バンクの場合と４バンクの場合について示す図である。
【図３５】２レベル分割の場合の１パスフィルタ動作を説明する図である。
【図３６】本発明のコンテキストモデルの一実施例を示すブロック図である。
【図３７】本発明の符号－絶対値フォーマッティングユニットの一実施例を示すブロック図である。
【図３８】本発明の絶対値メモリユニットの一実施例を示すブロック図である。
【図３９】本発明のシグニフィカンスユニットの一実施例を示すブロック図である。
【図４０】本発明のツリーメモリユニットの一実施例を示すブロック図である。
【図４１】ウエーブレット係数のアラインメントを説明するためのブロック図である。
【図４２】”１．５”アラインメントを使用した本発明のシグニフィカンスユニットの一実施例を示すブロック図である。
【図４３】１パス動作のための符号化データメモリの動的割り当ての例を示す図である。

70

【図４４】本発明の符号化プロセスの一実施例を示すフローチャートである。
【図４５】図４４に示したフローチャートの続きを示すフローチャートである。
【図４６】本発明の復号プロセスの一実施例を示すフローチャートである。
【図４７】図４６に示したフローチャートの続きを示すフローチャートである。
【図４８】本発明の符号化プロセス及び復号プロセスのための各係数のモデリングプロセスの一実施例を示すフローチャートである。
【図４９】図４８に示したフローチャートの続きを示すフローチャートである。
【図５０】本発明の符号化プロセスの他の実施例を示すフローチャートである。
【図５１】図５０に示したフローチャートの続きを示すフローチャートである。
【図５２】本発明の復号プロセスの他の実施例を示すフローチャートである。
【図５３】図５２に示したフローチャートの続きを示すフローチャートである。
【図５４】本発明の符号化プロセス及び復号プロセスにおける各係数のモデリングのためのプロセスの他の実施例を示すフローチャートである。
【図５５】図５４に示したフローチャートの続きを示すフローチャートである。
【図５６】本発明において係数アライメントのために用いられる周波数帯域用乗数の例を示す図である。
【図５７】１レベル分割ＲＴＳ変換用のフォワードフィルタを実現するためのＣ言語ソースリストの一例を示す図である。
【図５８】１レベル分割ＲＴＳ変換用のインバースフィルタを実現するためのＣ言語ソースリストの一例を示す図である。
【図５９】２レベル分割ＲＴＳ変換用のフォワードフィルタを実現するためのＣ言語ソースリストの一例を示す図である。
【図６０】２レベル分割ＲＴＳ変換用のインバースフィルタを実現するためのＣ言語ソースリストの一例を示す図である。
【符号の説明】
１０２　可逆ウエーブレット変換ブロック
１０３　ビット・シグニフィカンス埋め込みブロック
１０４　エントロピー符号化器
１０５　周波数ベースコンテキストモデル
１０６　統合空間／周波数（ＪＳＦ）コンテキストモデル
１０８　スイッチ
１０９　符号－絶対値フォーマッティングユニット
４０１，４０５，４０９　低域通過フィルタユニット

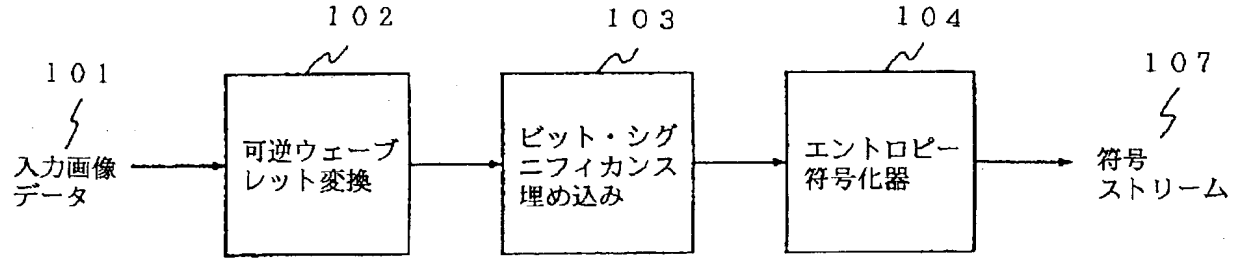
71

４０２，４０６，４１０　高域通過フィルタユニット
４０３，４０４，４０７　サブサンプリングユニット
４０８，４１１，４１２　サブサンプリングユニット
１０００　フォワード・ウエーブレットフィルタ
１００１，１００６　加算器
１００２，１００５　減算器
１００３　１ビットシフトブロック
１００４　２ビットシフトブロック
１００７　２ビットシフトブロック
１１０１　減算器
１１０２　２ビットシフトブロック
１１０３　１ビットシフトブロック
１１０４　減算器
１１０５　加算器
１１０６　減算器
１１０７，１１０８　クリッピングブロック
１３０１　フィルタ制御ユニット
１３０２　メモリ
１３０３　フィルタ
１４０１　メモリインターフェイスユニット
１７００　コンテキストモデル
１７０１　絶対値メモリ

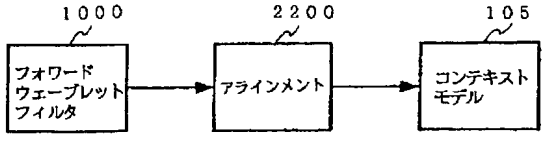
72

１７０２　シグニフィカンスユニット
１７０３　ツリーメモリ
１８０１　インバータ
１８０２　マルチプレクサ
１８０３　プライオリティエンコーダ
１８０４　カウンタ
１９００～１９１６　カウンタ
１９２０～１９３６　メモリ
２００１　インデックスカウンタ
２００２　一致論理
２００３　インデックスーレベル論理
２００４　シグニフィカンスカウンタ
２００５～２００７　フリップフロップ（フラグ０～２）
２００８　デシジョンジェネレータ
２１０１　メモリ
２１０２　カウンタ
２１０３　マルチプレクサ
２１１０～２１１２　レジスタ
２２００　アラインメントユニット
２３０１　”１．５”プライオリティエンコーダ
２３０２　マルチプレクサ

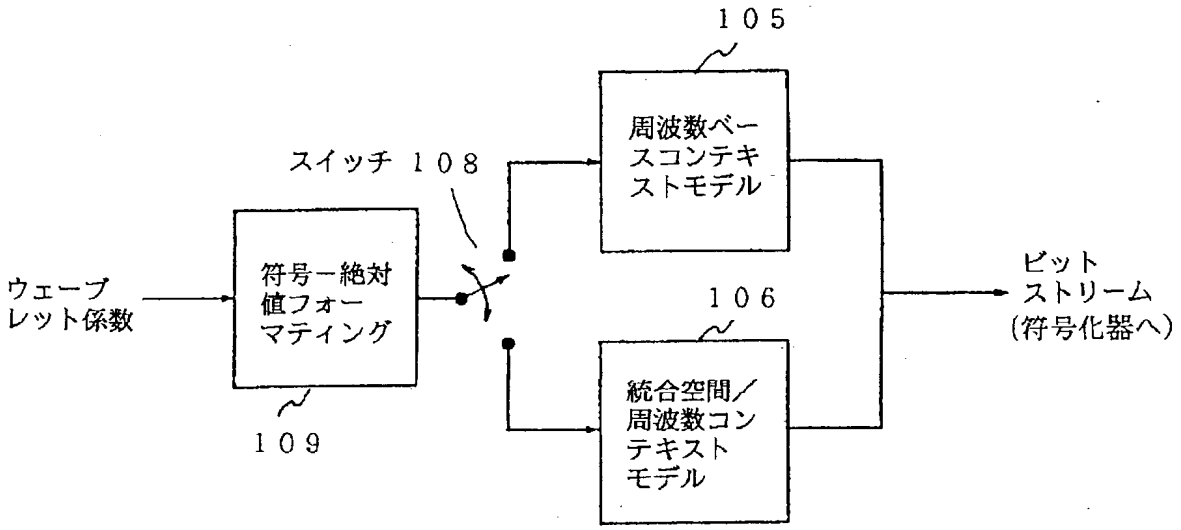
【図１】



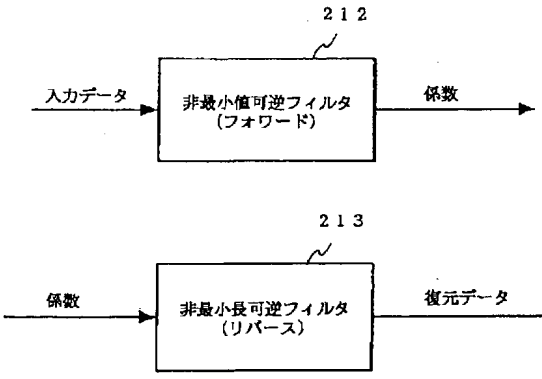
【図４１】



【図２】

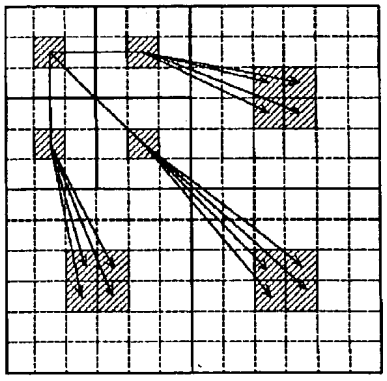


【図４】

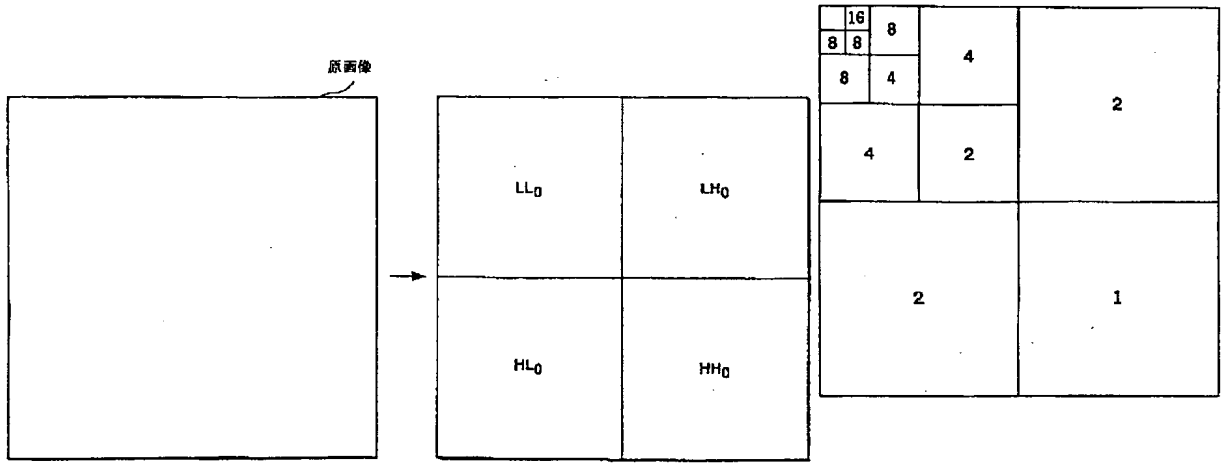


【図５】

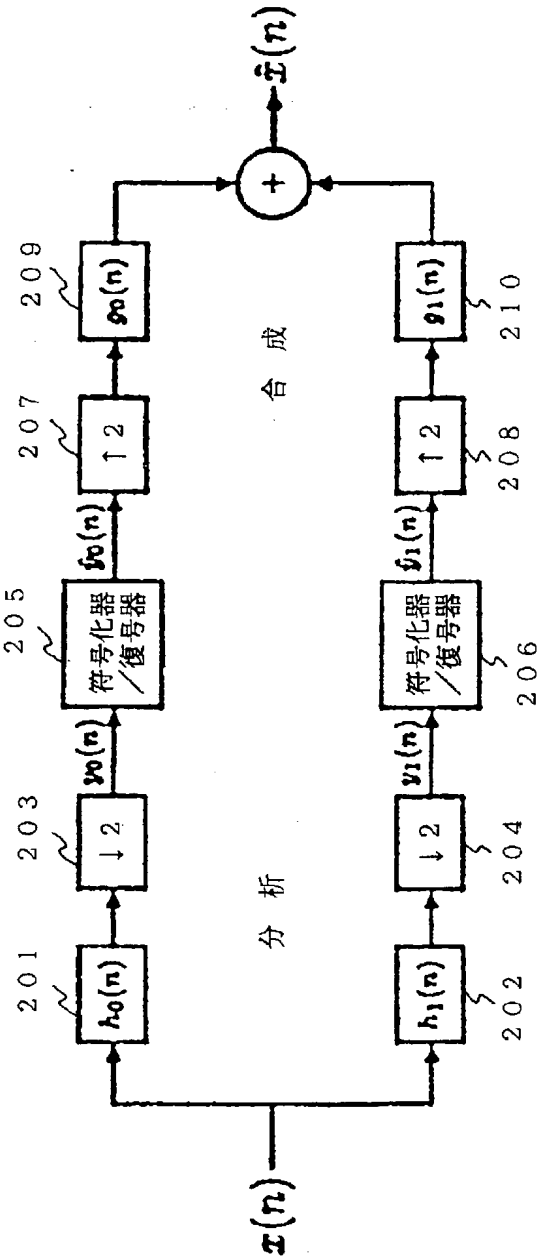
【図１４】



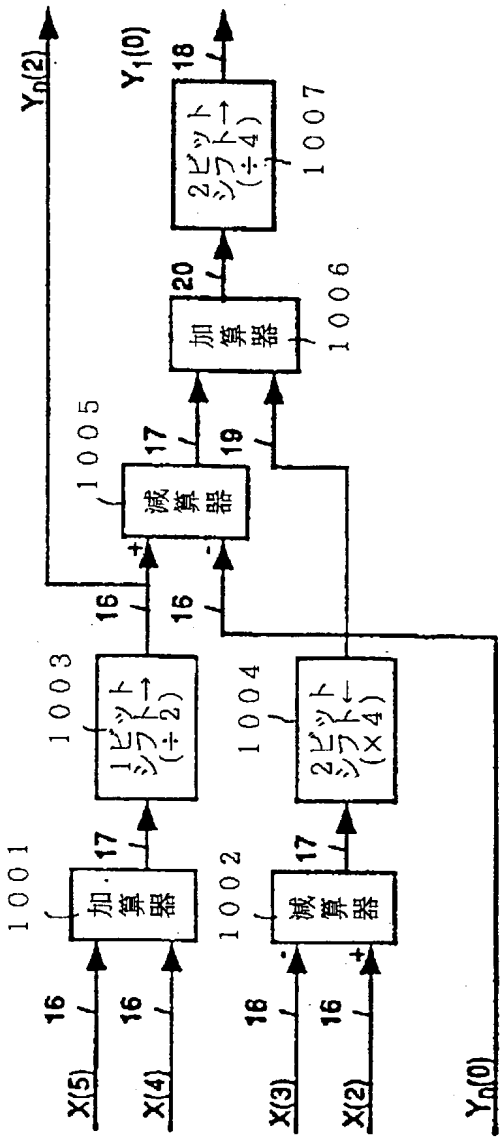
【図５６】



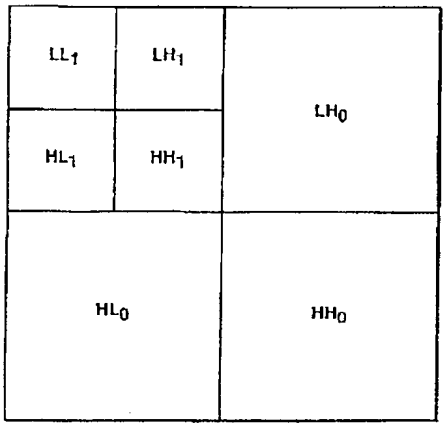
【図３】


x(n)
x̂(n)
分析
合成
h0(n)
h1(n)
↓2
↑2
y0(n)
y1(n)
符号化器／復号器
201
202
203
204
205
206
207
208
209
210

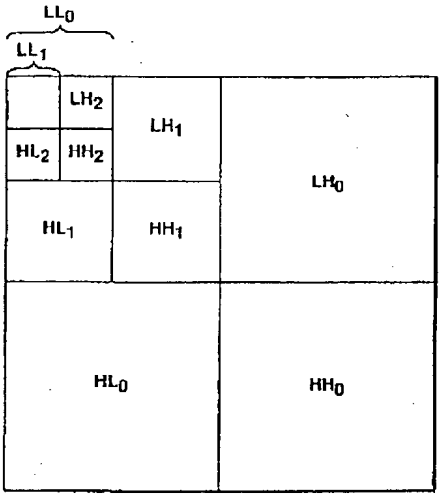
【図２９】


Y0(2)
Y1(0)
Y0(0)
X(5)
X(4)
X(3)
X(2)
加算器
減算器
1ビットシフト→(÷2)
2ビットシフト←(×4)
2ビットシフト→(÷4)
1001
1002
1003
1004
1005
1006
1007
16
17
18
19
20

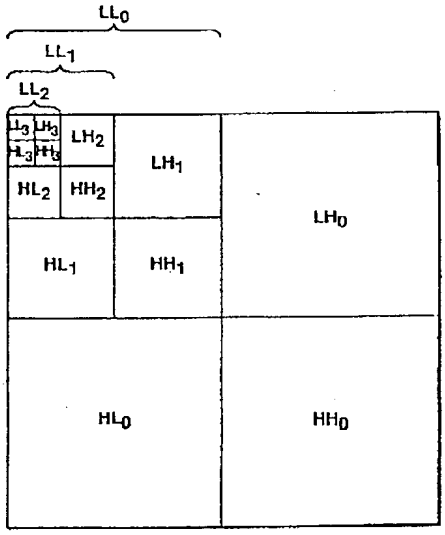
【図６】



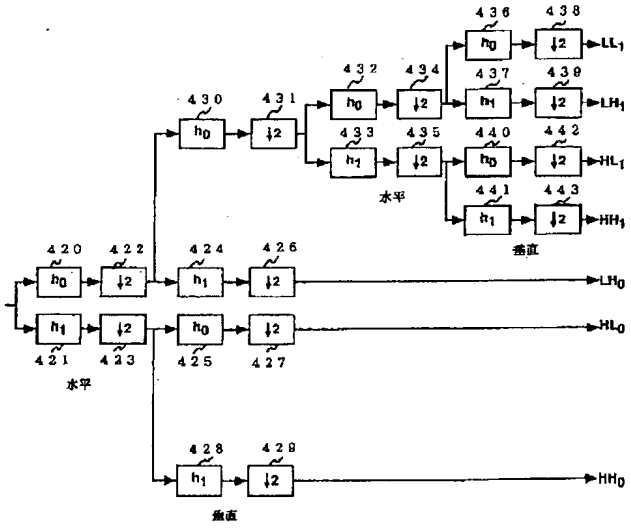
【図７】



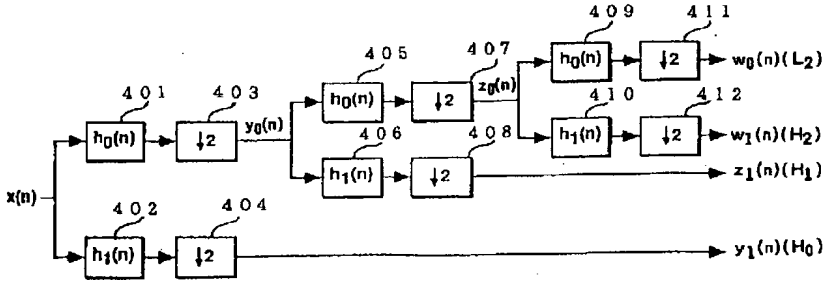
【図８】



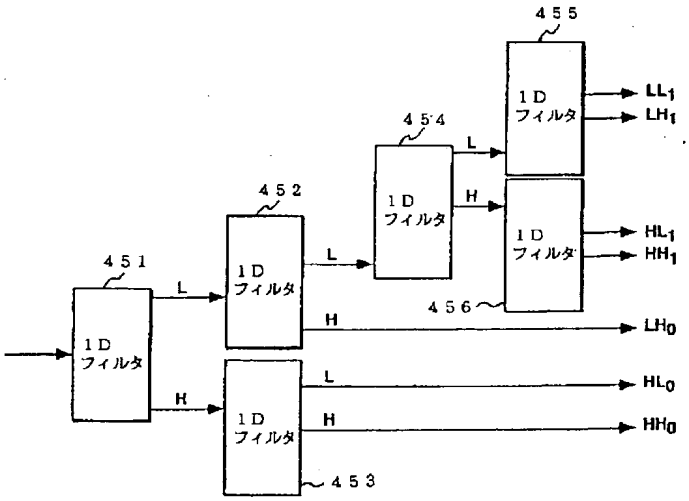
【図１０】



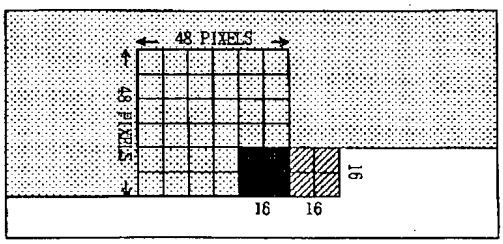
【図９】



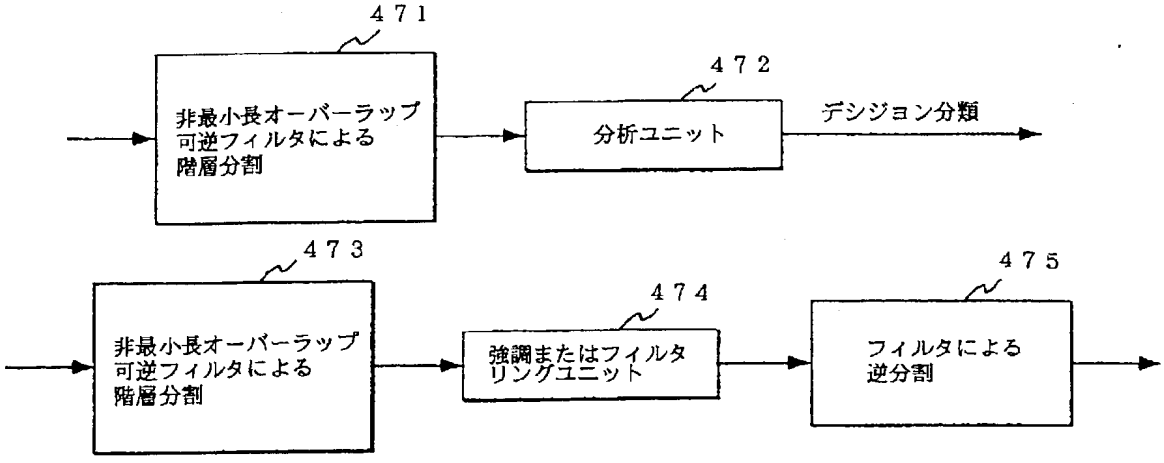
【図１１】


455
1D フィルタ
LL1
LH1
454
L
1D フィルタ
H
452
1D フィルタ
HL1
HH1
L
451
1D フィルタ
456
H
LH0
L
1D フィルタ
HL0
H
HH0
453

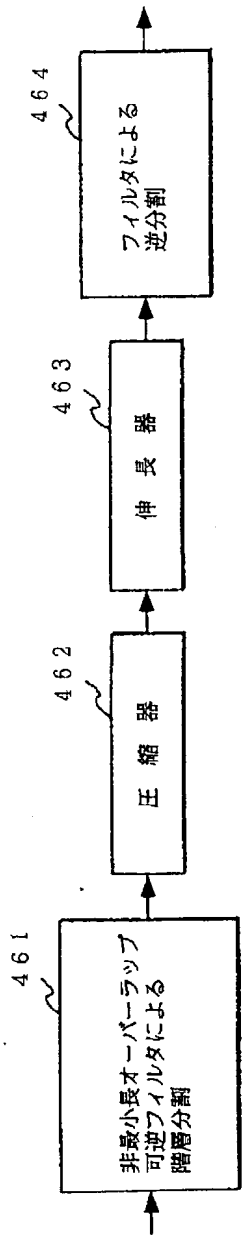
【図３１】


48 PIXELS
48 PIXELS
16
16
16

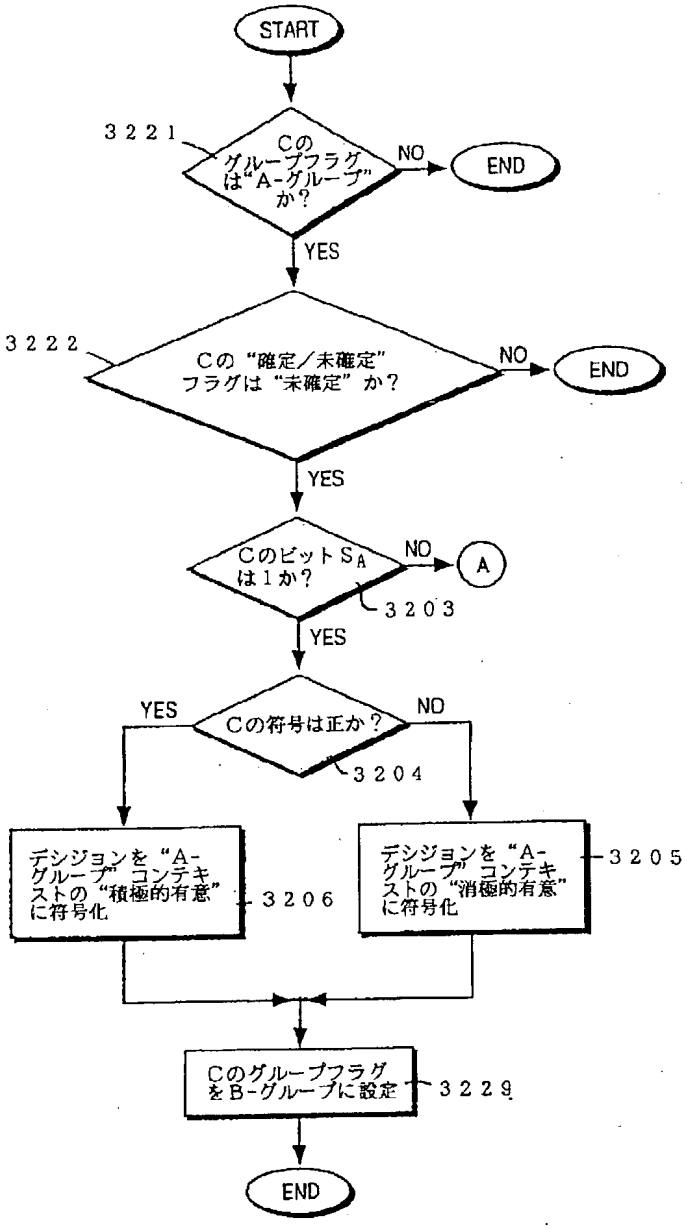
【図１３】


471
非最小長オーバーラップ
可逆フィルタによる
階層分割
472
分析ユニット
デシジョン分類
473
非最小長オーバーラップ
可逆フィルタによる
階層分割
474
強調またはフィルタ
リングユニット
475
フィルタによる
逆分割

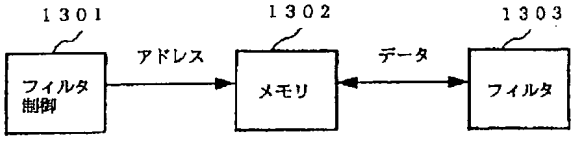
【図１２】


461
非最小長オーバーラップ
可逆フィルタによる
階層分割
462
圧縮器
463
伸長器
464
フィルタによる
逆分割

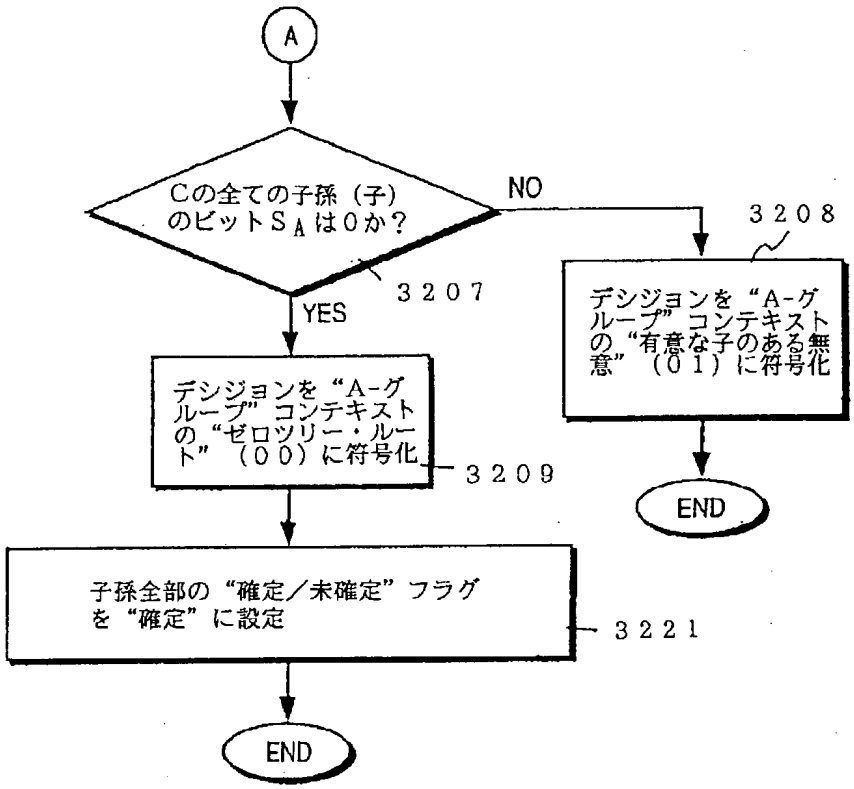
【図１５】


START
3221
Cの
グループフラグ
は"A-グループ"
か?
NO
END
YES
3222
Cの"確定/未確定"
フラグは"未確定"か?
NO
END
YES
CのビットSA
は1か?
NO
A
3203
YES
YES
Cの符号は正か?
NO
3204
デシジョンを"A-
グループ"コンテキ
ストの"積極的有意"
に符号化
3206
デシジョンを"A-
グループ"コンテキ
ストの"消極的有意"
に符号化
3205
Cのグループフラグ
をB-グループに設定
3229
END

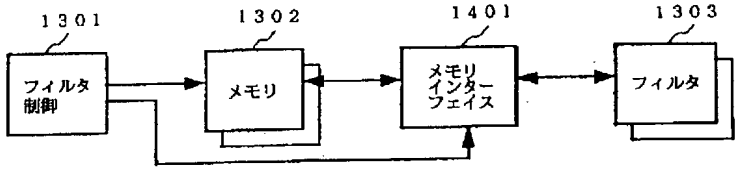
【図３２】


1301
フィルタ
制御
アドレス
1302
メモリ
データ
1303
フィルタ

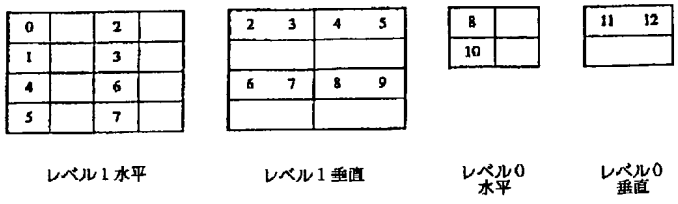
【図１６】


A
Cの全ての子孫（子）のビットSAは0か？
NO
3208
3207
YES
デシジョンを“A-グループ”コンテキストの“有意な子のある無意”（01）に符号化
デシジョンを“A-グループ”コンテキストの“ゼロツリー・ルート”（00）に符号化
3209
END
子孫全部の“確定／未確定”フラグを“確定”に設定
3221
END

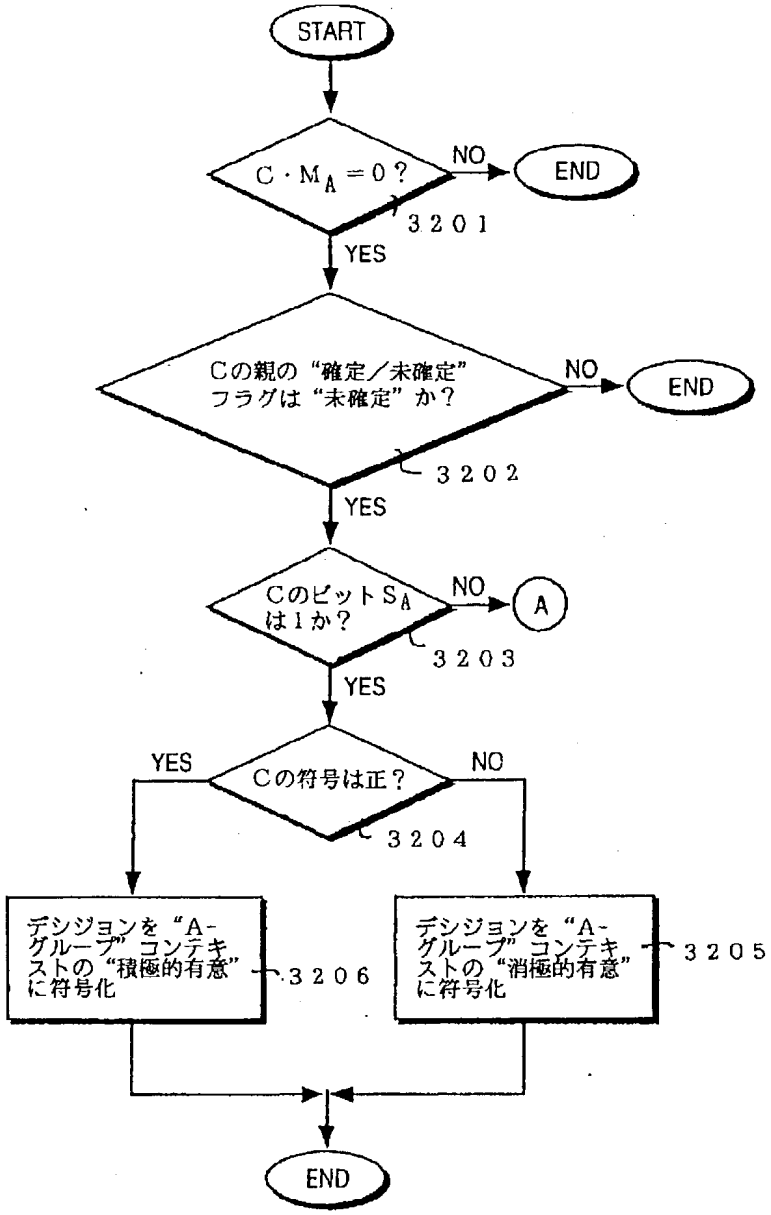
【図３３】


1301
フィルタ制御
1302
メモリ
1401
メモリインターフェイス
1303
フィルタ

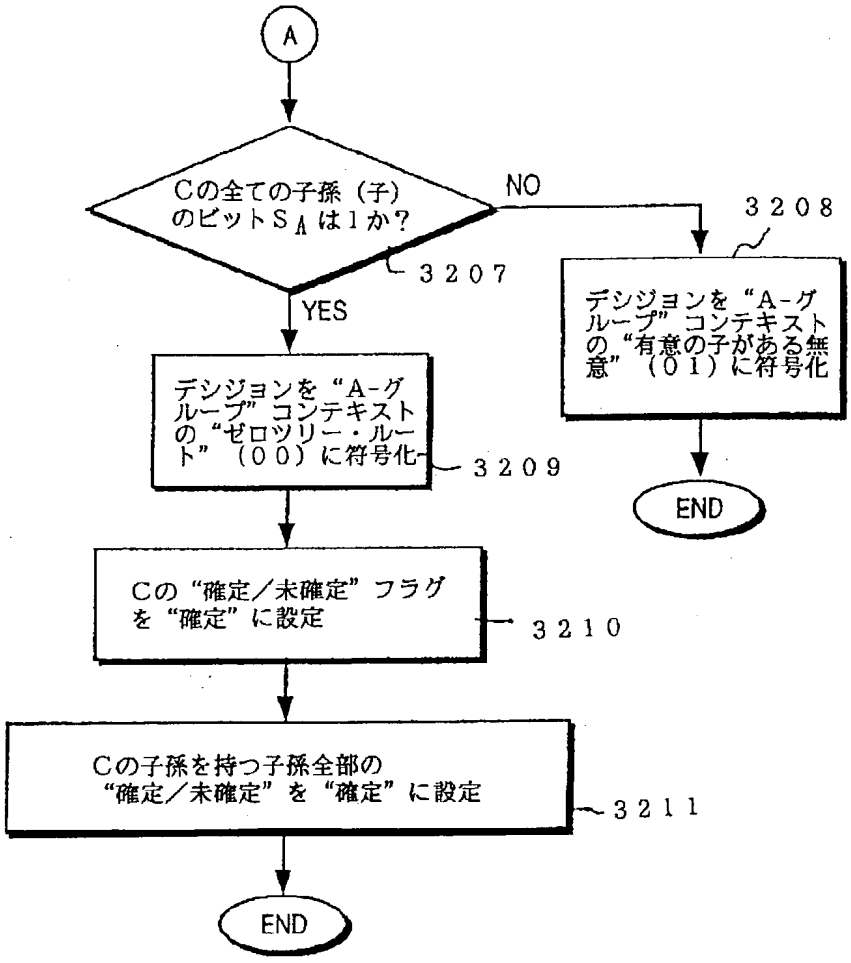
【図３５】


0 2
1 3
4 6
5 7
レベル1水平
2 3 4 5
6 7 8 9
レベル1垂直
8
10
レベル0水平
11 12
レベル0垂直

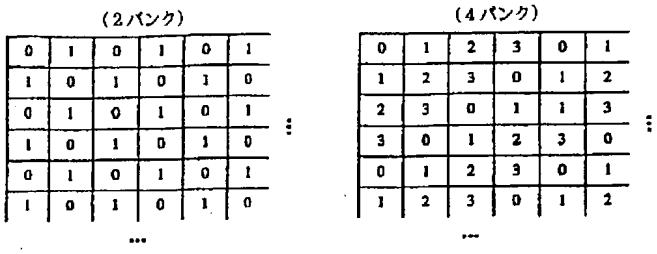
【図１７】


START
C・M_A＝0？
NO
END
3201
YES
Cの親の“確定／未確定”フラグは“未確定”か？
NO
END
3202
YES
CのビットS_Aは1か？
NO
A
3203
YES
YES
Cの符号は正？
NO
3204
デシジョンを“A-グループ”コンテキストの“積極的有意”に符号化
3206
デシジョンを“A-グループ”コンテキストの“消極的有意”に符号化
3205
END

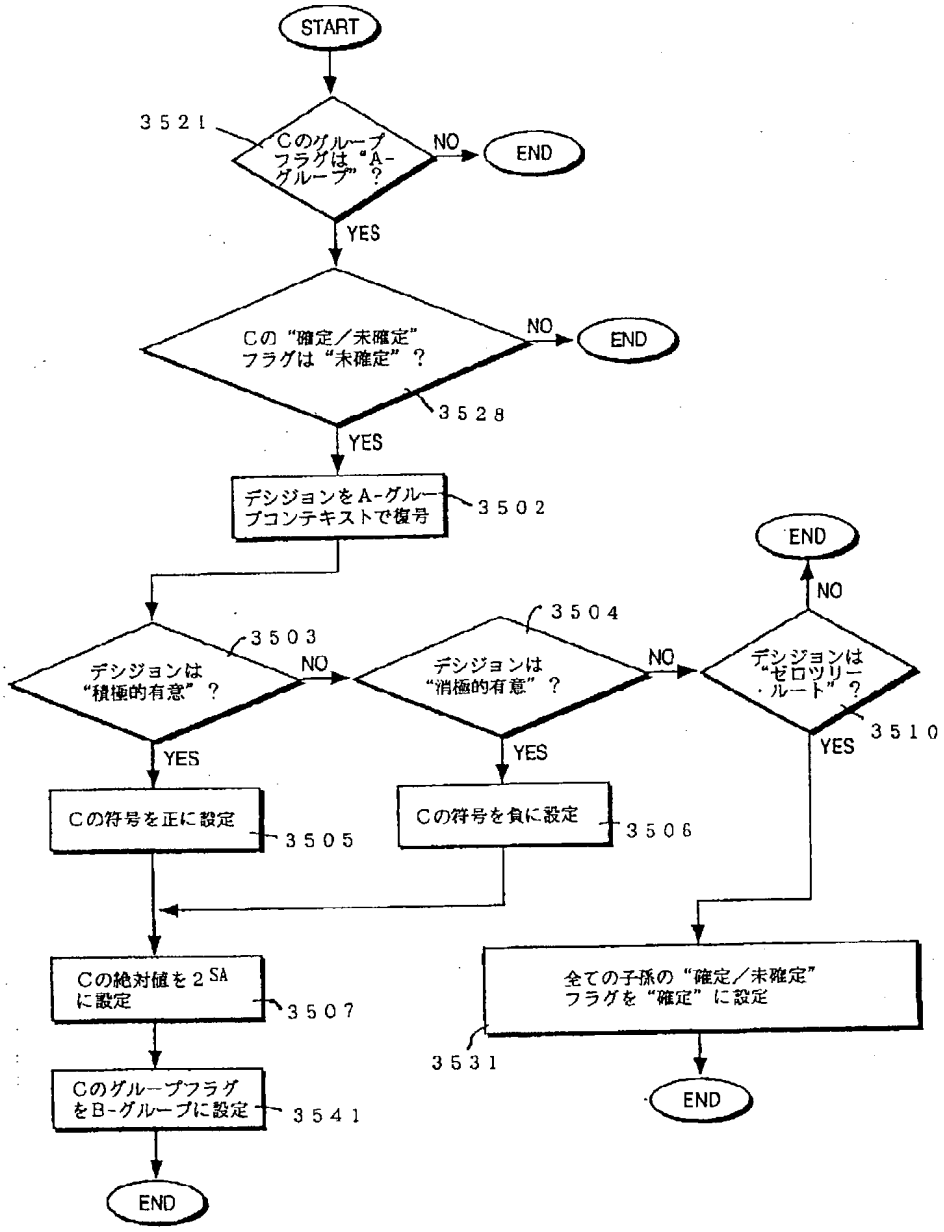
【図１８】


A
Cの全ての子孫（子）のビットS_Aは1か？
NO
3208
3207
YES
デシジョンを“A-グループ”コンテキストの“有意の子がある無意”（０１）に符号化
デシジョンを“A-グループ”コンテキストの“ゼロツリー・ルート”（００）に符号化
3209
END
Cの“確定／未確定”フラグを“確定”に設定
3210
Cの子孫を持つ子孫全部の“確定／未確定”を“確定”に設定
3211
END

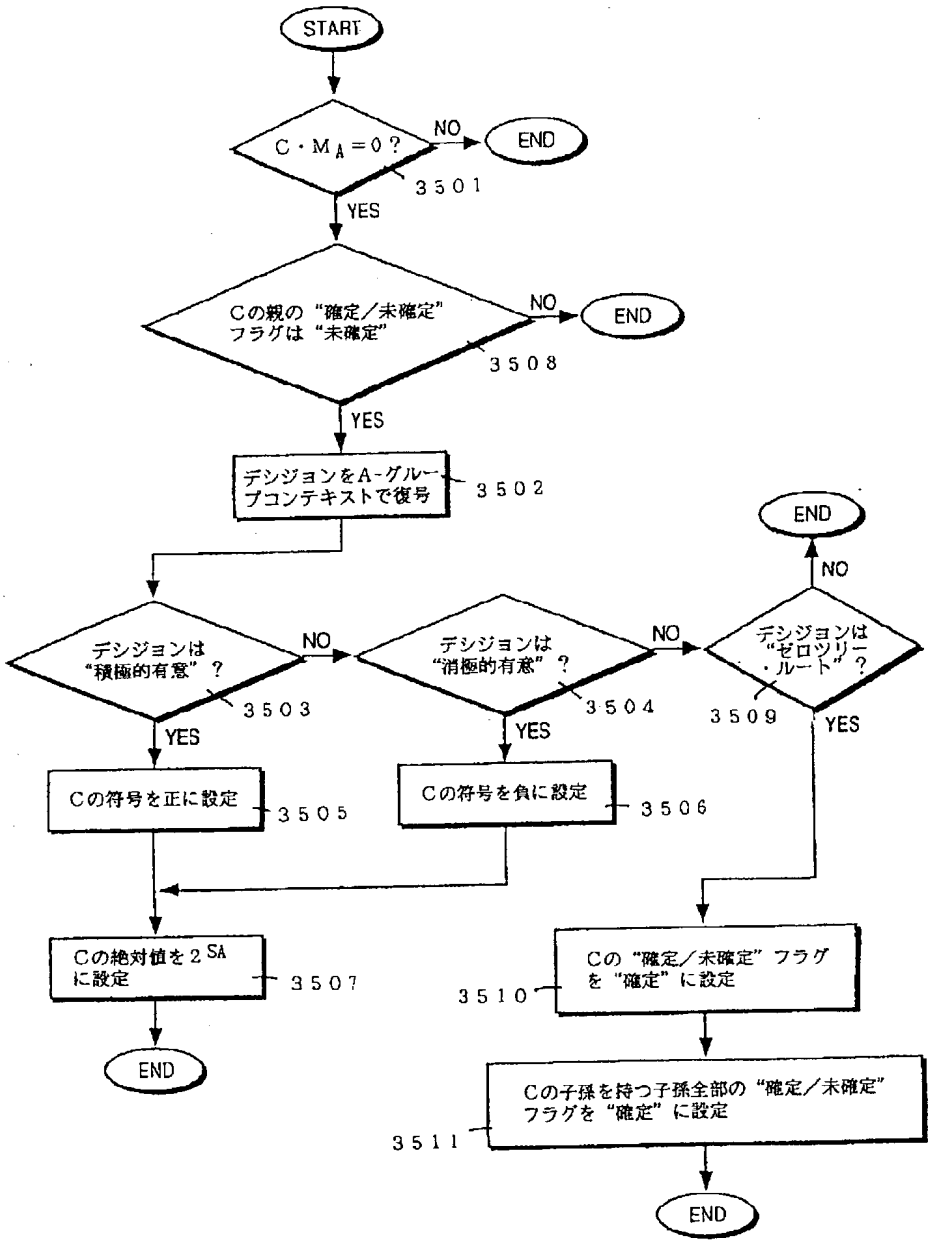
【図３４】


(2バンク)
(4バンク)

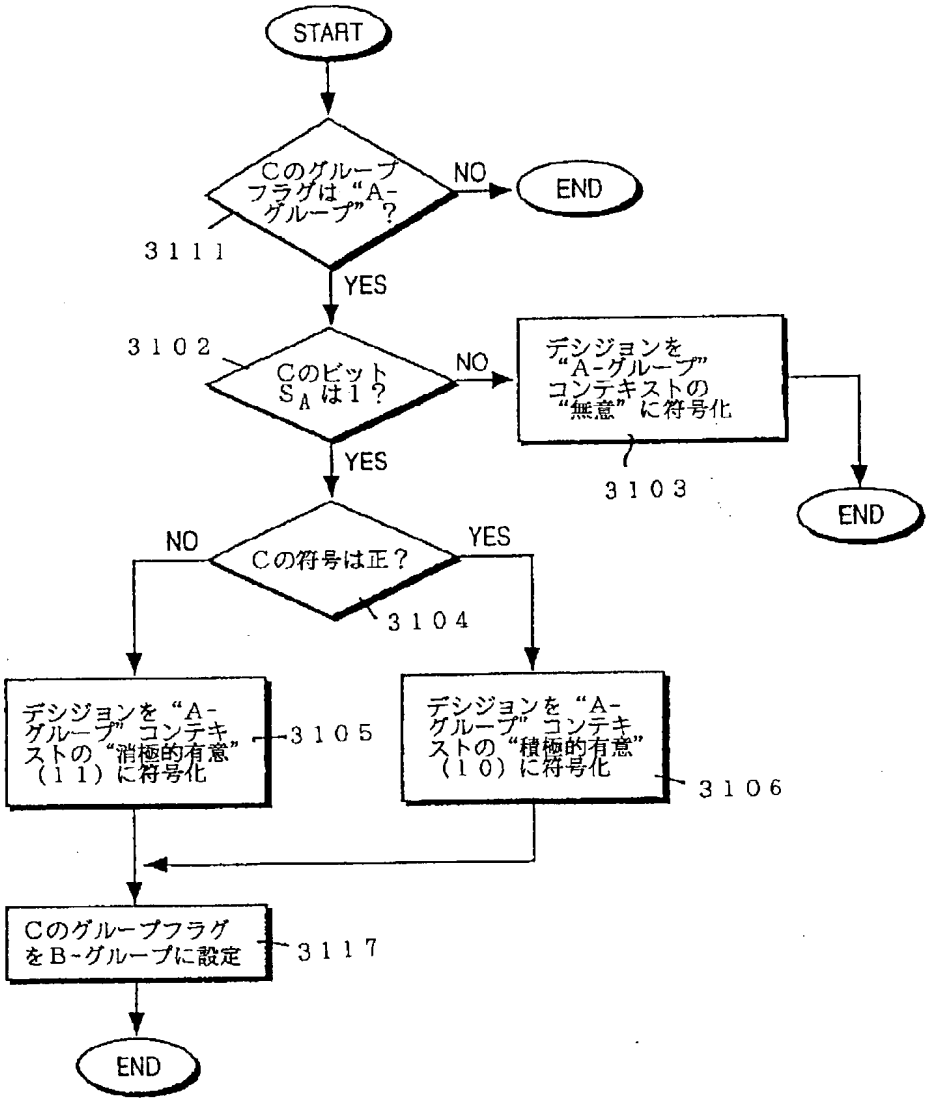
【図１９】


START
3521
Cのグループフラグは“A-グループ”？
NO
END
YES
Cの“確定／未確定”フラグは“未確定”？
NO
END
3528
YES
デシジョンをA-グループコンテキストで復号
3502
3503
デシジョンは“積極的有意”？
NO
3504
デシジョンは“消極的有意”？
NO
デシジョンは“ゼロツリー・ルート”？
3510
END
NO
YES
YES
YES
Cの符号を正に設定
3505
Cの符号を負に設定
3506
Cの絶対値を2^SAに設定
3507
Cのグループフラグをb-グループに設定
3541
END
全ての子孫の“確定／未確定”フラグを“確定”に設定
3531
END

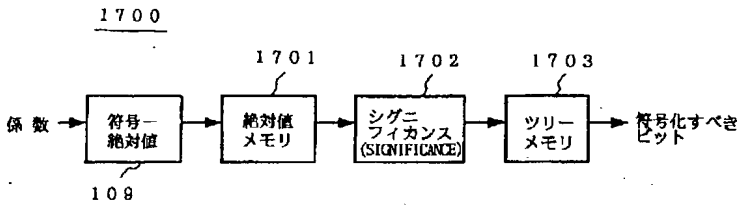
【図２０】


START
C・M_A＝0？
NO
END
3501
YES
Cの親の“確定／未確定”フラグは“未確定”
NO
END
3508
YES
デシジョンをA-グループコンテキストで復号
3502
デシジョンは“積極的有意”？
NO
3503
YES
デシジョンは“消極的有意”？
NO
3504
YES
デシジョンは“ゼロツリー・ルート”？
3509
YES
NO
END
Cの符号を正に設定
3505
Cの符号を負に設定
3506
Cの絶対値を2^SAに設定
3507
END
Cの“確定／未確定”フラグを“確定”に設定
3510
Cの子孫を持つ子孫全部の“確定／未確定”フラグを“確定”に設定
3511
END

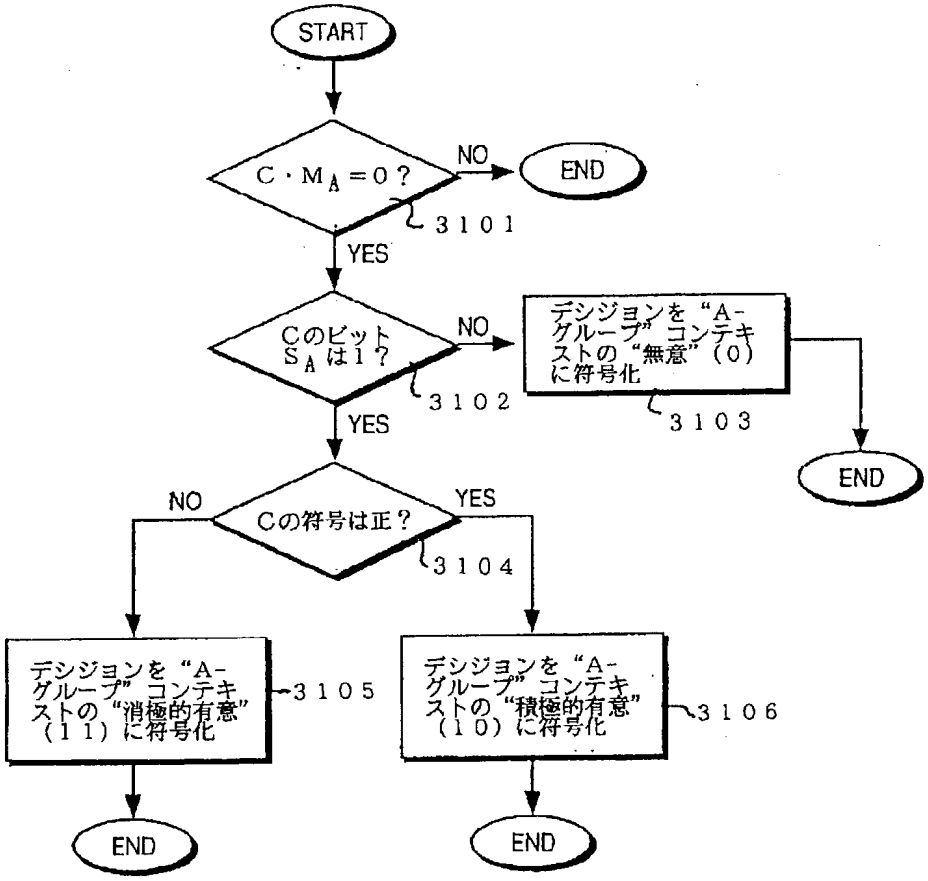
【図２１】


START
Cのグループフラグは“A-グループ”？
NO
END
3111
YES
3102
Cのビット S_A は１？
NO
デシジョンを“A-グループ”コンテキストの“無意”に符号化
3103
END
YES
NO
Cの符号は正？
YES
3104
デシジョンを“A-グループ”コンテキストの“消極的有意”（１１）に符号化
3105
デシジョンを“A-グループ”コンテキストの“積極的有意”（１０）に符号化
3106
Cのグループフラグをｂ-グループに設定
3117
END

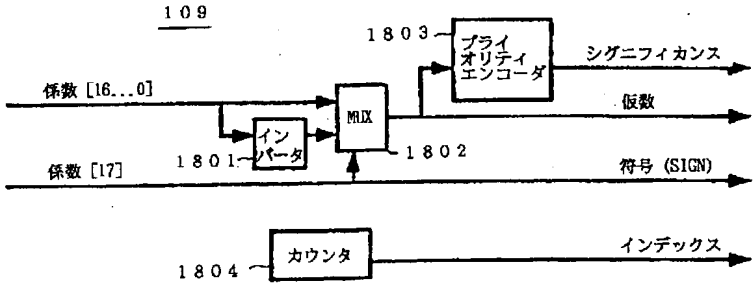
【図３６】


1700
1701
1702
1703
係数
符号－絶対値
109
絶対値メモリ
シグニフィカンス(SIGNIFICANCE)
ツリーメモリ
符号化すべきビット

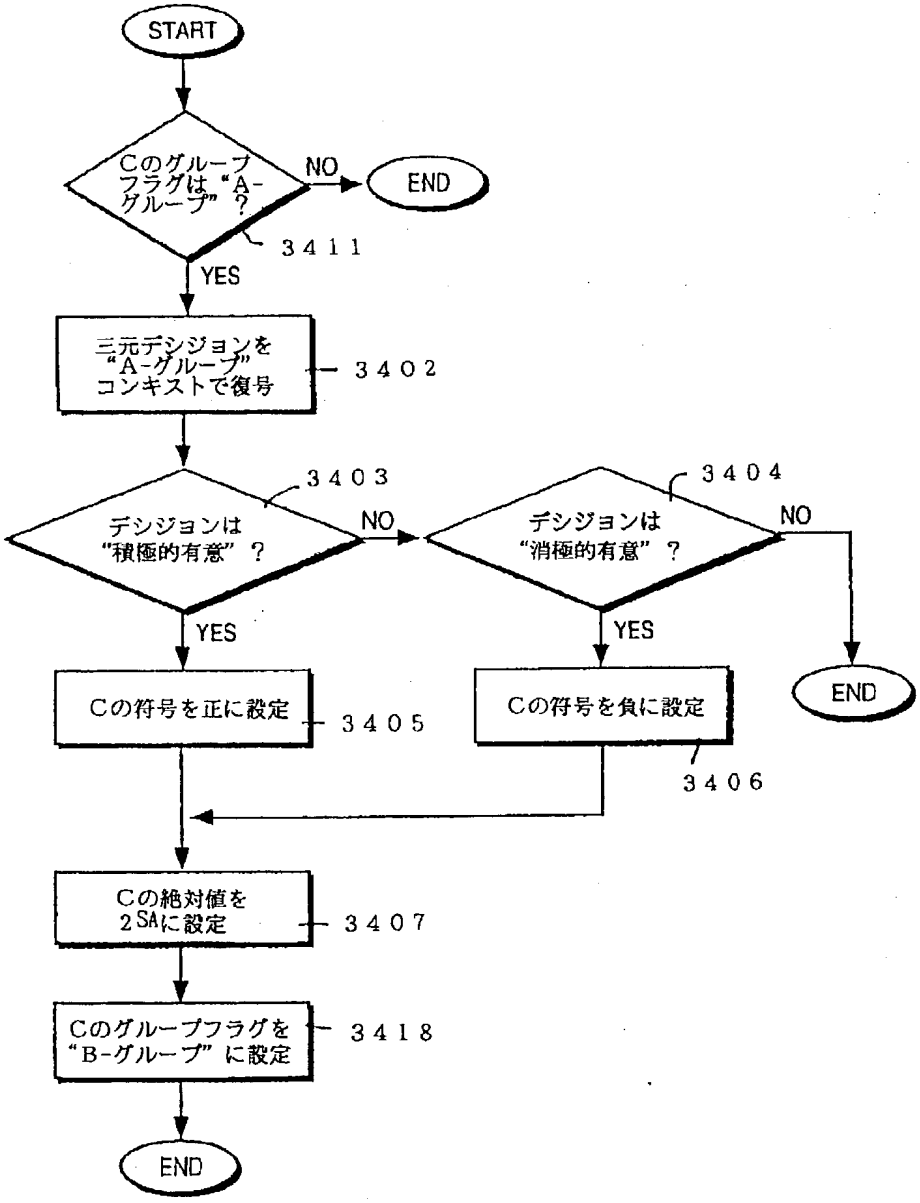
【図２２】


START
C・M_A＝０？
NO
END
３１０１
YES
Cのビット S_A は１？
NO
デシジョンを“A-グループ”コンテキストの“無意”（０）に符号化
３１０２
３１０３
YES
END
NO
Cの符号は正？
YES
３１０４
デシジョンを“A-グループ”コンテキストの“消極的有意”（１１）に符号化
３１０５
デシジョンを“A-グループ”コンテキストの“積極的有意”（１０）に符号化
３１０６
END
END

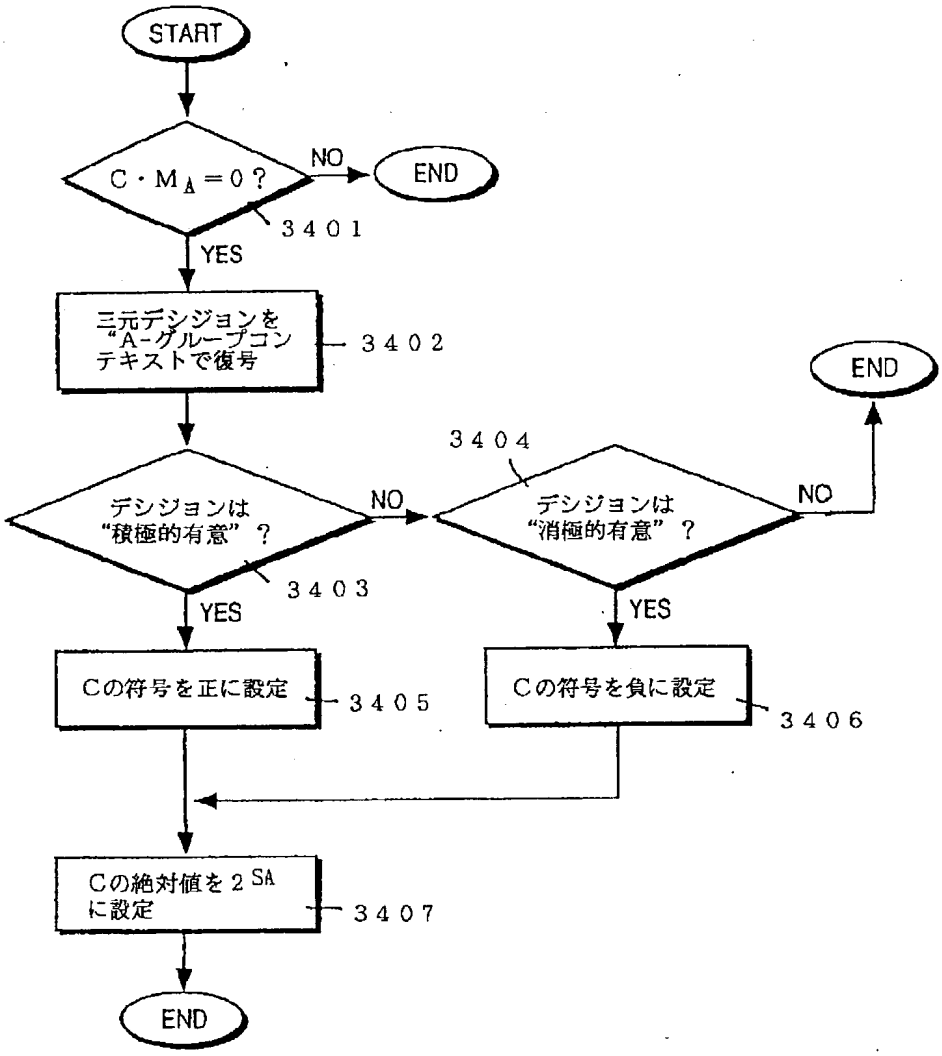
【図３７】


１０９
１８０３
プライオリティエンコーダ
シグニフィカンス
係数 [16...0]
仮数
MUX
インバータ
１８０１
１８０２
係数 [17]
符号 (SIGN)
カウンタ
１８０４
インデックス

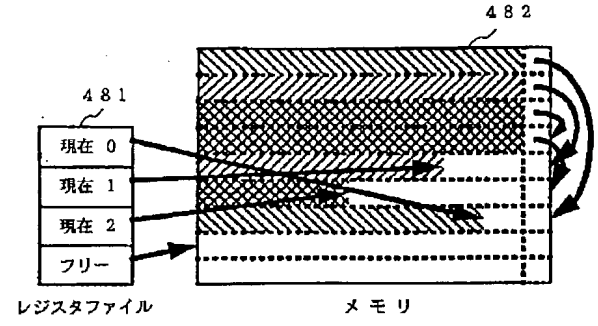
【図２３】


START
Cのグループフラグは“A-グループ”？
NO
END
3411
YES
三元デシジョンを“A-グループ”コンキストで復号
3402
3403
デシジョンは“積極的有意”？
NO
3404
デシジョンは“消極的有意”？
NO
YES
YES
Cの符号を正に設定
3405
Cの符号を負に設定
END
3406
Cの絶対値を2SAに設定
3407
Cのグループフラグを“B-グループ”に設定
3418
END

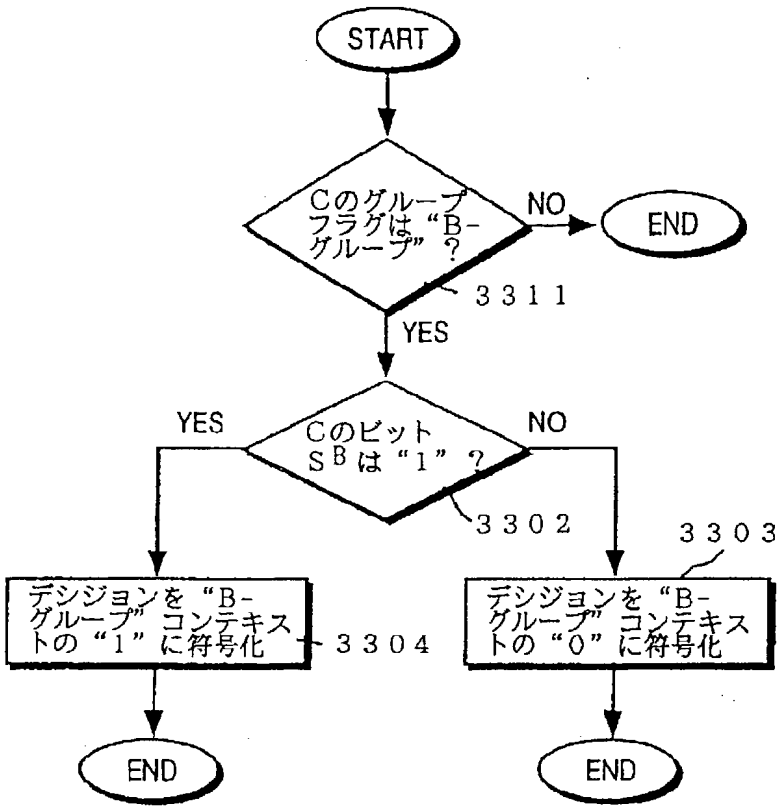
【図２４】


START
C・M_A＝0？
NO
END
3401
YES
三元デシジョンを“A-グループコンテキストで復号
3402
END
3404
デシジョンは“積極的有意”？
NO
デシジョンは“消極的有意”？
NO
3403
YES
YES
Cの符号を正に設定
3405
Cの符号を負に設定
3406
Cの絶対値を2^SAに設定
3407
END

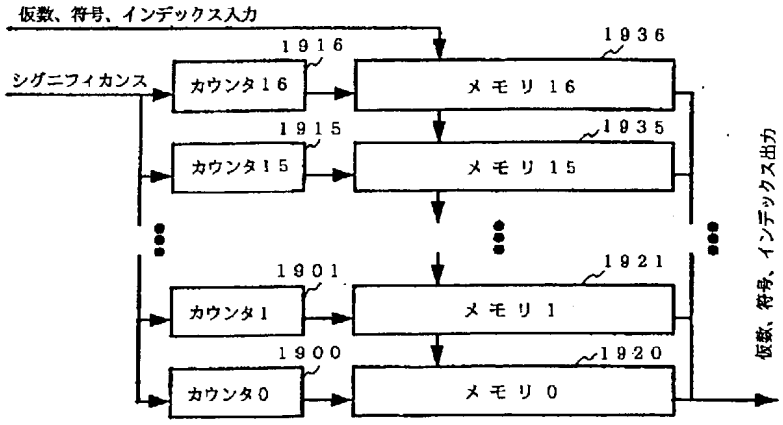
【図４３】


482
481
現在 0
現在 1
現在 2
フリー
レジスタファイル
メモリ

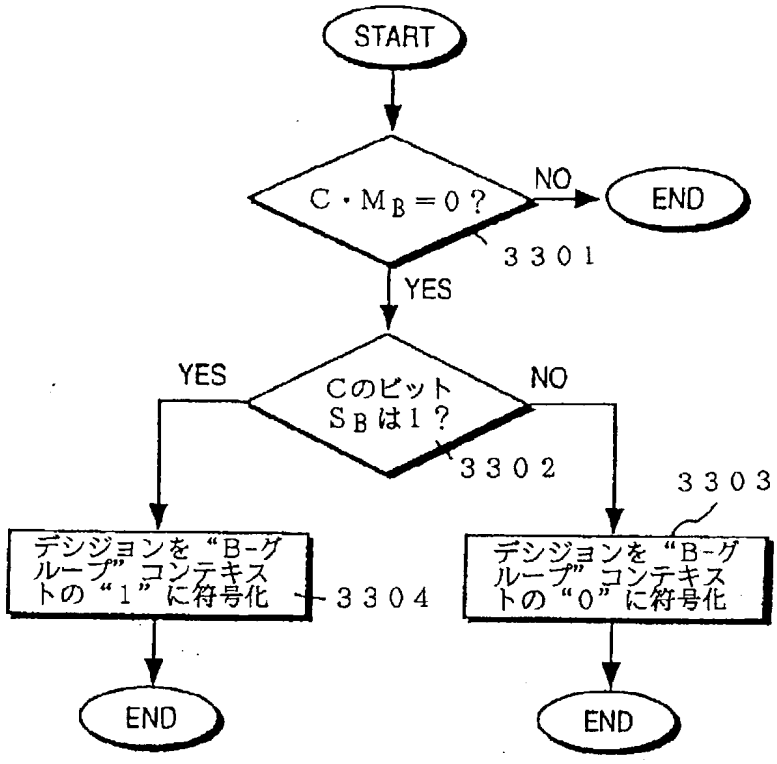
【図２５】


START
Cのグループフラグは“B-グループ”？
NO
END
3311
YES
YES
CのビットSBは“1”？
NO
3302
3303
デシジョンを“B-グループ”コンテキストの“1”に符号化
3304
デシジョンを“B-グループ”コンテキストの“0”に符号化
END
END

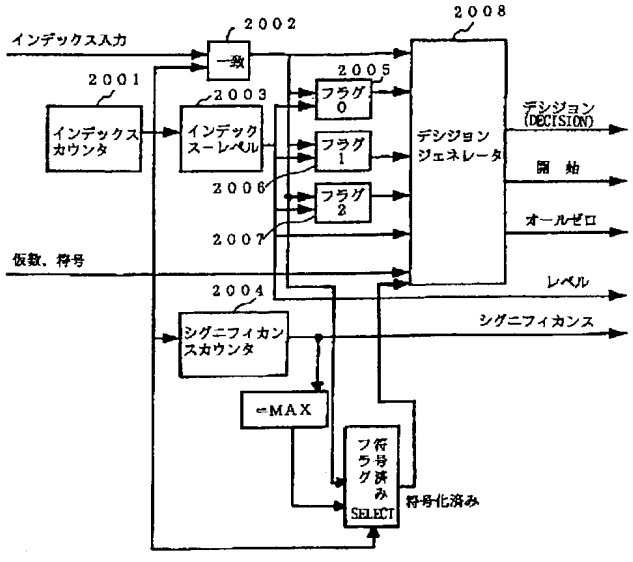
【図３８】


仮数、符号、インデックス入力
シグニフィカンス
1916
カウンタ16
メモリ16
1936
1915
カウンタ15
メモリ15
1935
1901
カウンタ1
メモリ1
1921
1900
カウンタ0
メモリ0
1920
仮数、符号、インデックス出力

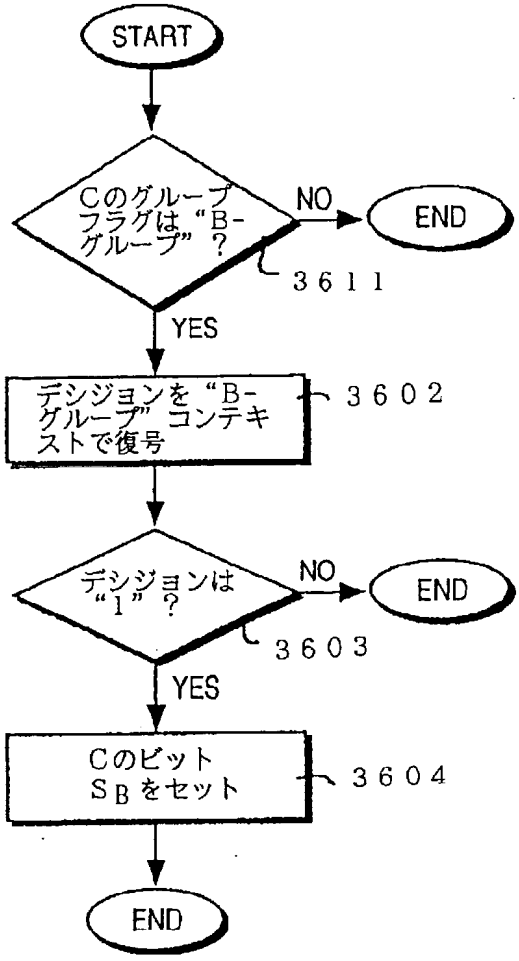
【図２６】


START
C・M_B＝0？
NO
END
3301
YES
YES
Cのビット
S_B は1？
NO
3302
3303
デシジョンを“B-グループ”コンテキストの“1”に符号化
3304
デシジョンを“B-グループ”コンテキストの“0”に符号化
END
END

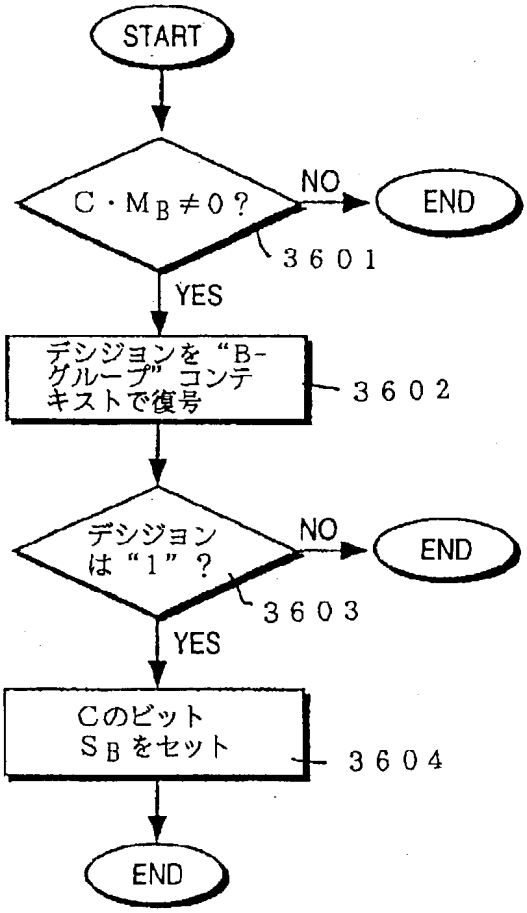
【図３９】


インデックス入力
2002
一致
2008
2001
2005
フラグ0
インデックスカウンタ
2003
インデックスレベル
フラグ1
デシジョンジェネレータ
デシジョン(DECISION)
2006
フラグ2
開始
オールゼロ
2007
仮数、符号
レベル
2004
シグニフィカンスカウンタ
シグニフィカンス
MAX
符号済みフラグ
SELECT
符号化済み

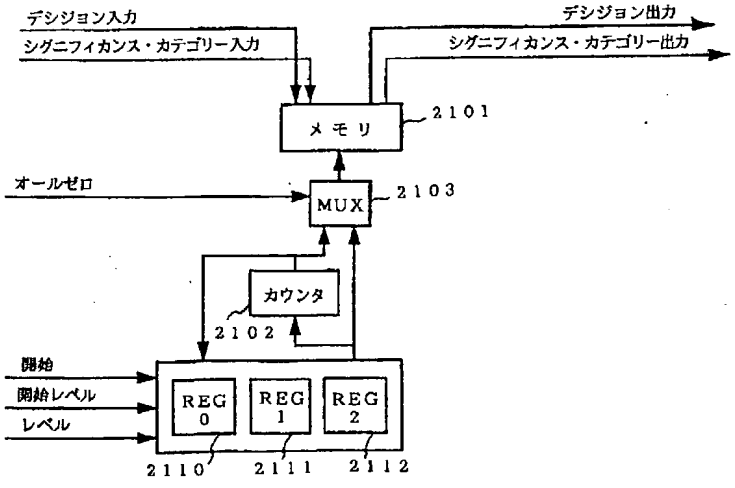
【図２７】


START
Cのグループフラグは“B-グループ”？
NO
END
3611
YES
デシジョンを“B-グループ”コンテキストで復号
3602
デシジョンは“1”？
NO
END
3603
YES
Cのビット S_B をセット
3604
END

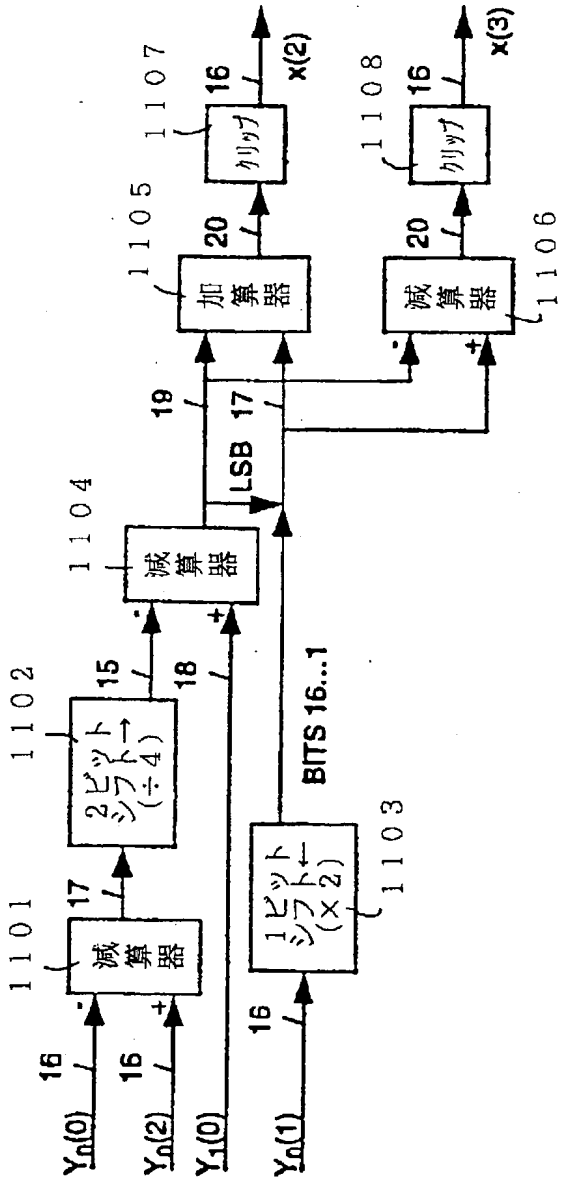
【図２８】


START
C・$M_B \neq 0$？
NO
END
3601
YES
デシジョンを“B-グループ”コンテキストで復号
3602
デシジョンは“1”？
NO
END
3603
YES
Cのビット S_B をセット
3604
END

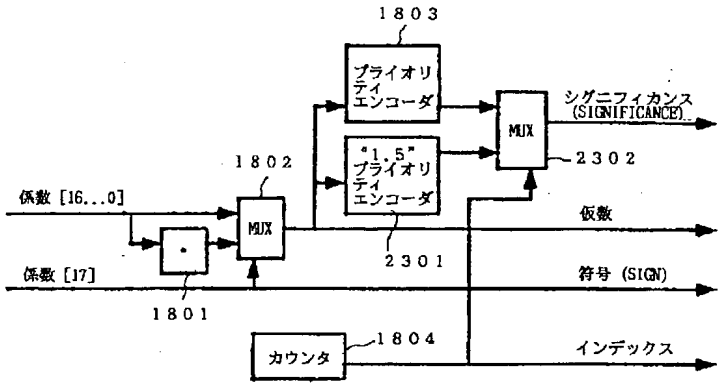
【図４０】


デシジョン入力
シグニフィカンス・カテゴリー入力
デシジョン出力
シグニフィカンス・カテゴリー出力
メモリ
2101
オールゼロ
MUX
2103
カウンタ
2102
開始
開始レベル
レベル
REG 0
REG 1
REG 2
2110
2111
2112

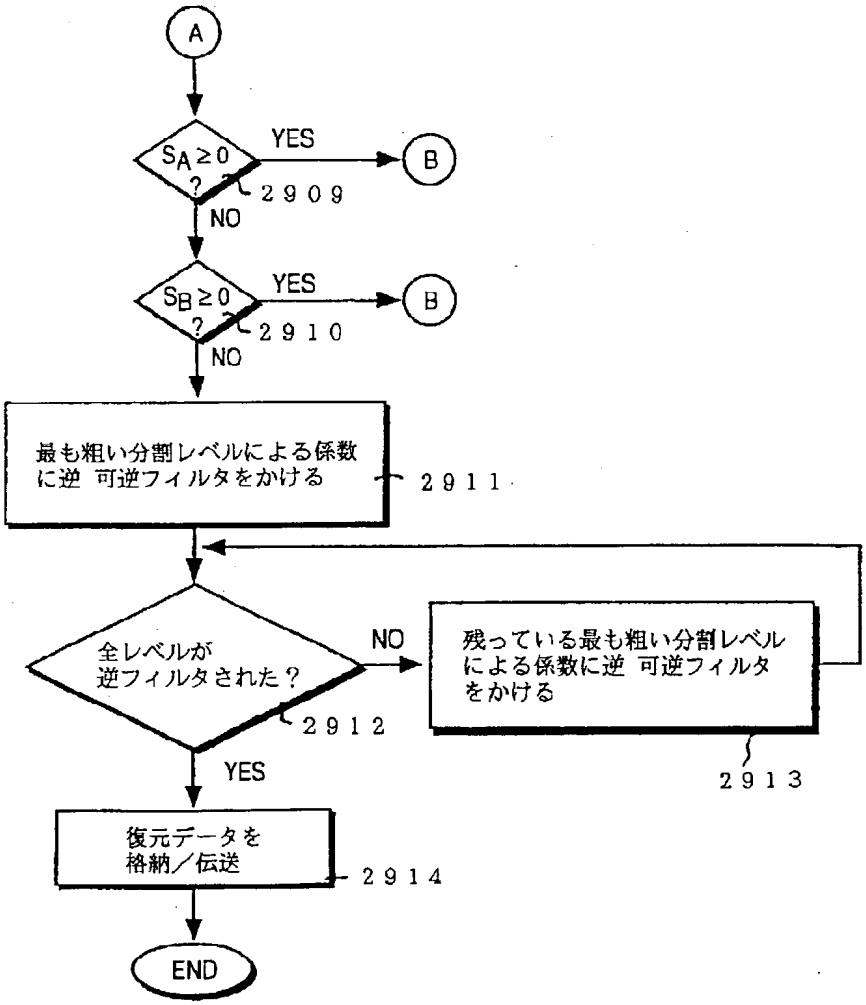
【図30】


減算器
1101
2ビット→
(÷4)
1102
減算器
1104
1ビット←
(×2)
1103
加算器
1105
減算器
1106
クリップ
1107
クリップ
1108
LSB
BITS 16...1
16
17
15
18
19
20
16
16
16
20
16
Yn(0)
Yn(2)
Y1(0)
Yn(1)
x(2)
x(3)

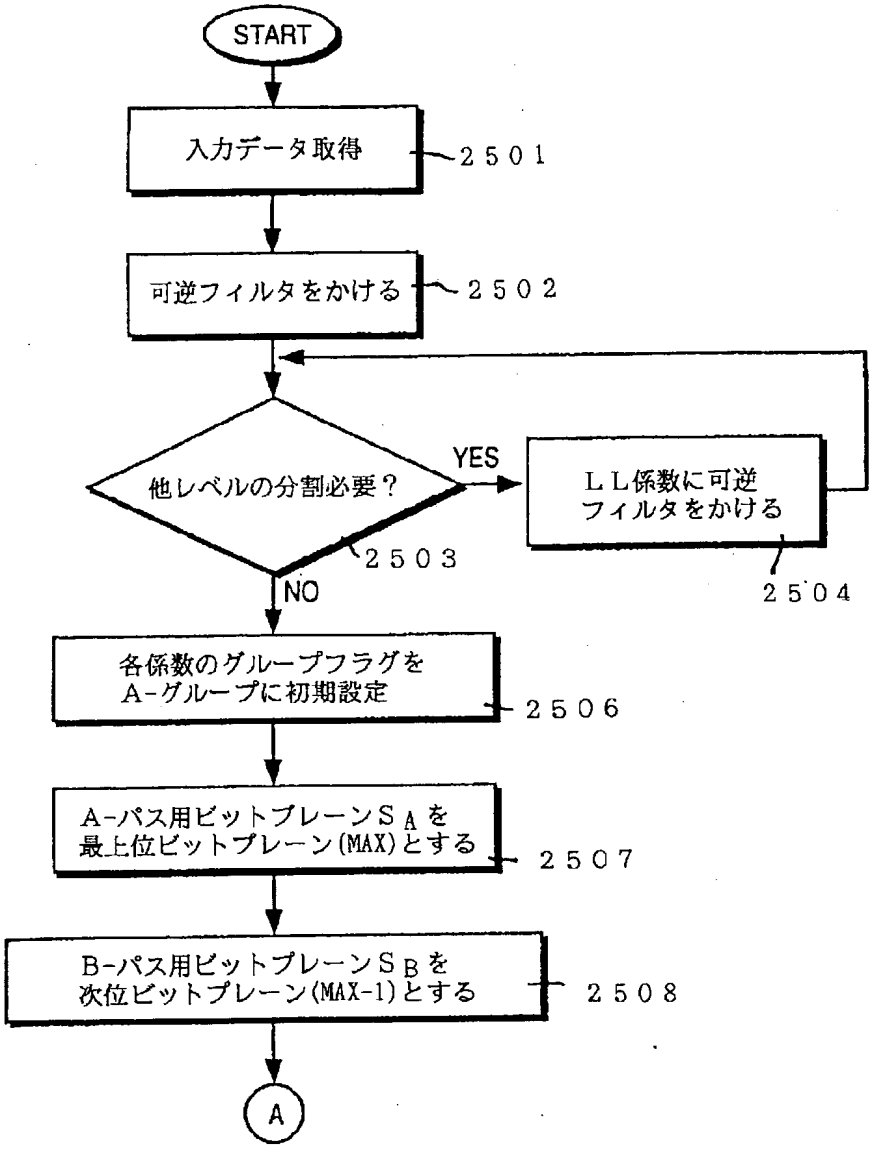
【図４２】


1803
プライオリティエンコーダ
"1.5"プライオリティエンコーダ
MUX
シグニフィカンス(SIGNIFICANCE)
2302
1802
係数[16...0]
MUX
仮数
2301
係数[17]
符号(SIGN)
1801
カウンタ
1804
インデックス

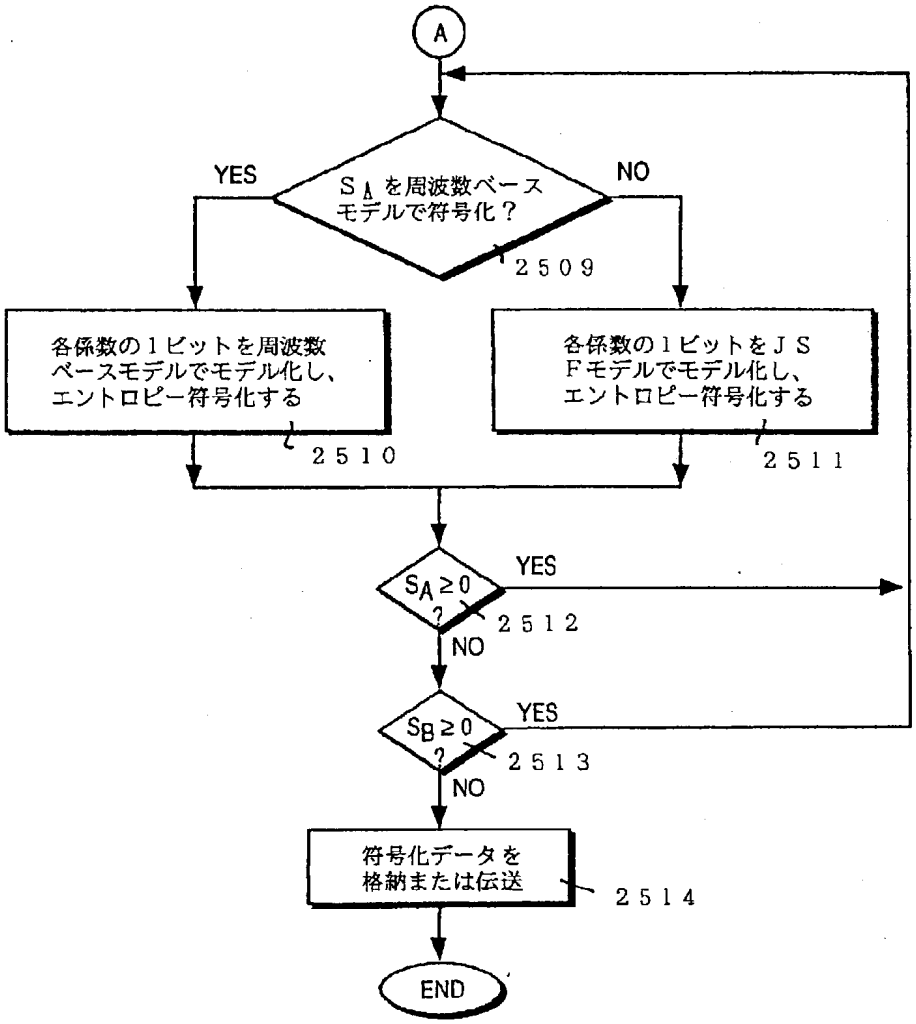
【図５３】


A
SA≥0 ?
YES
B
2909
NO
SB≥0 ?
YES
B
2910
NO
最も粗い分割レベルによる係数に逆 可逆フィルタをかける
2911
全レベルが逆フィルタされた？
NO
残っている最も粗い分割レベルによる係数に逆 可逆フィルタをかける
2912
2913
YES
復元データを格納／伝送
2914
END

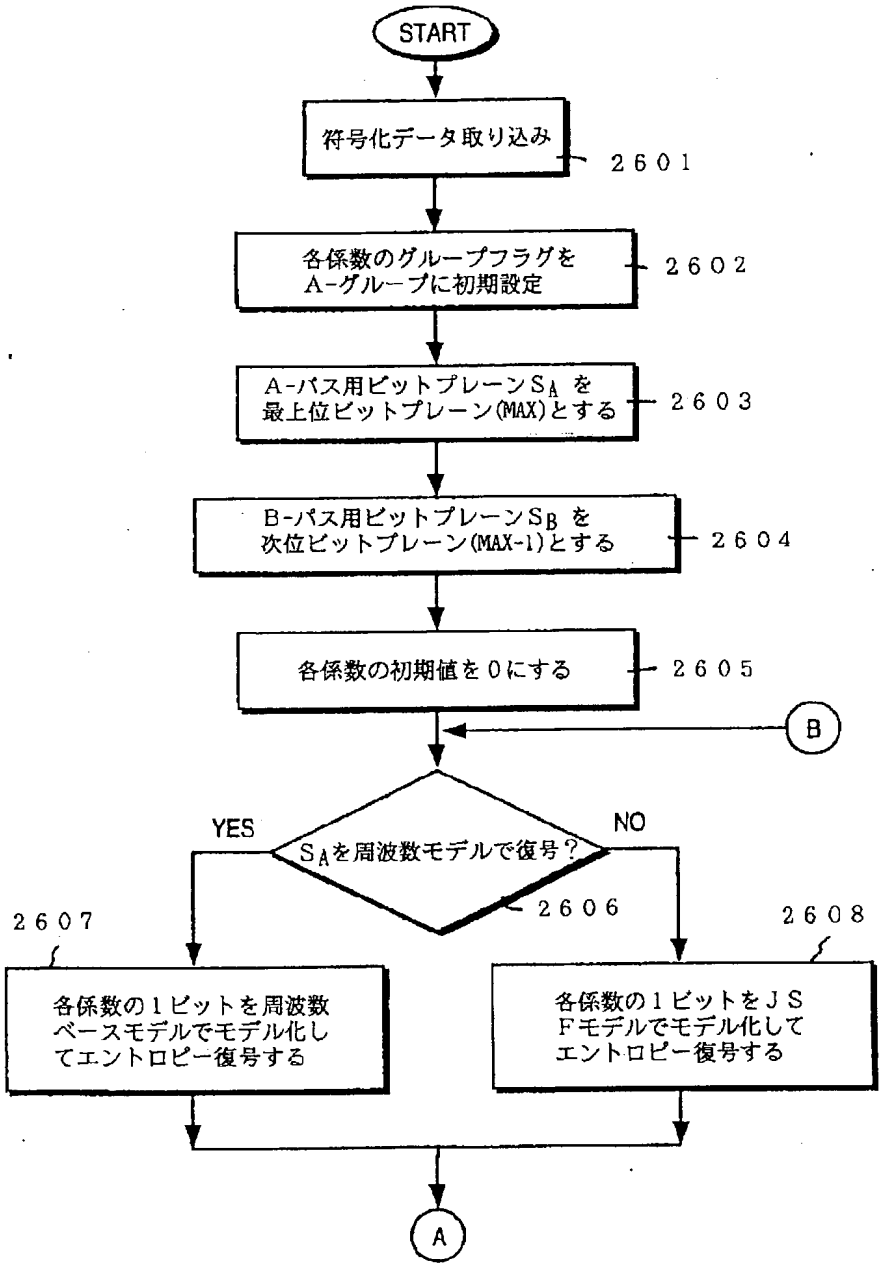
【図４４】


START
入力データ取得
２５０１
可逆フィルタをかける
２５０２
他レベルの分割必要？
YES
２５０３
ＬＬ係数に可逆
フィルタをかける
２５０４
NO
各係数のグループフラグを
A-グループに初期設定
２５０６
A-パス用ビットプレーンＳＡを
最上位ビットプレーン(MAX)とする
２５０７
B-パス用ビットプレーンＳＢを
次位ビットプレーン(MAX-1)とする
２５０８
A

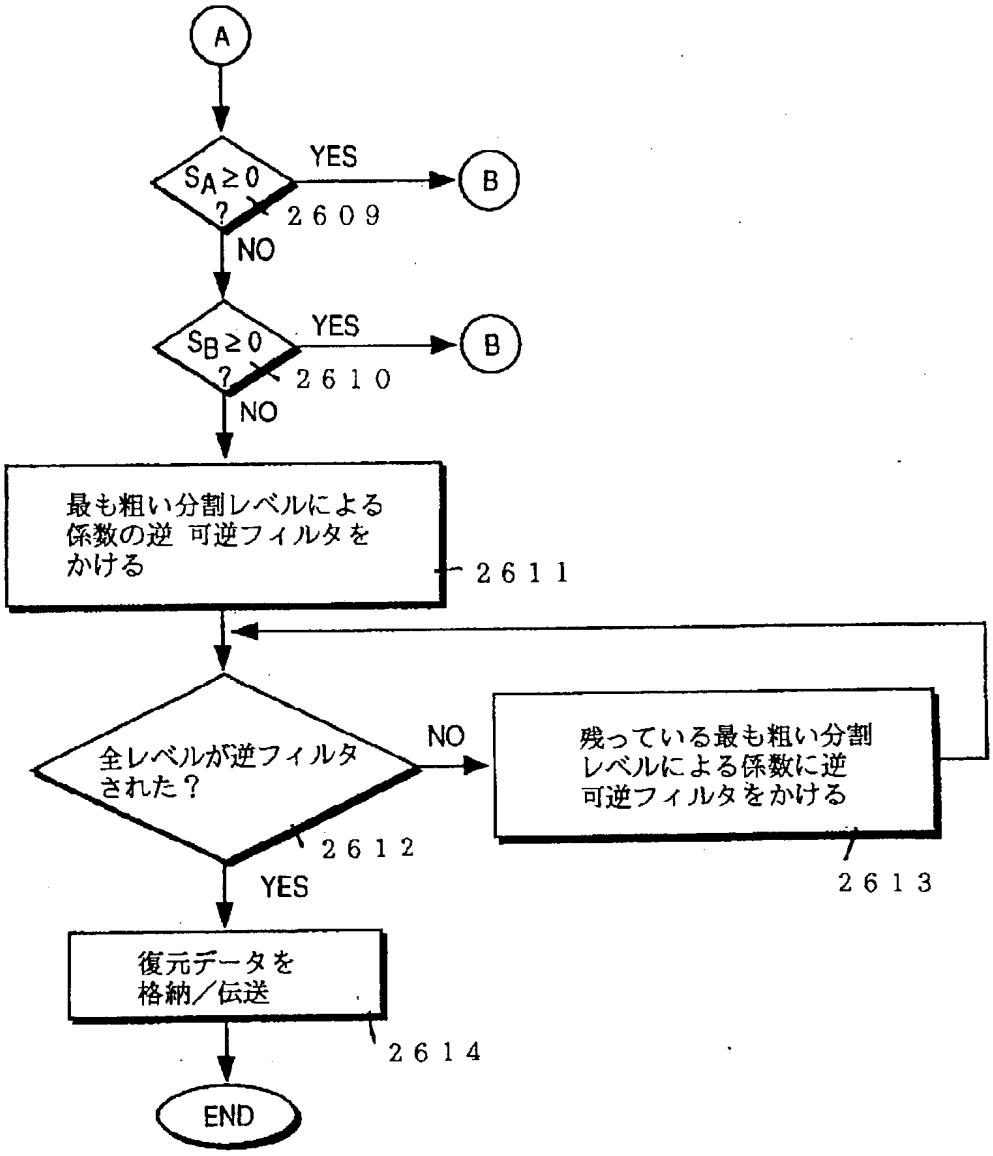
【図４５】


A
YES
S_Aを周波数ベースモデルで符号化？
NO
2509
各係数の１ビットを周波数ベースモデルでモデル化し、エントロピー符号化する
2510
各係数の１ビットをＪＳＦモデルでモデル化し、エントロピー符号化する
2511
$S_A \geq 0$?
YES
2512
NO
$S_B \geq 0$?
YES
2513
NO
符号化データを格納または伝送
2514
END

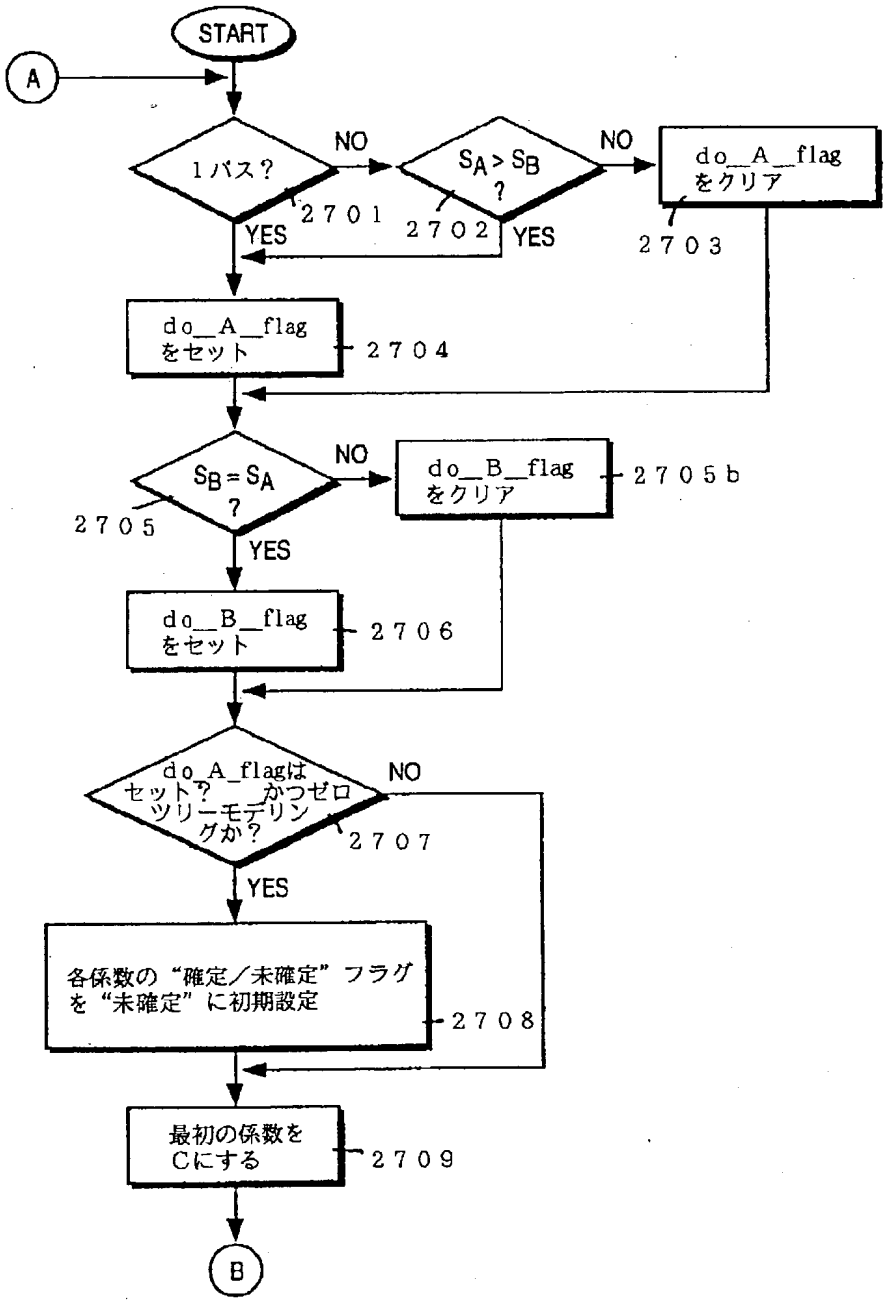
【図４６】


START
符号化データ取り込み
2601
各係数のグループフラグを
A-グループに初期設定
2602
A-パス用ビットプレーンS_A を
最上位ビットプレーン(MAX)とする
2603
B-パス用ビットプレーンS_B を
次位ビットプレーン(MAX-1)とする
2604
各係数の初期値を０にする
2605
B
YES
S_Aを周波数モデルで復号？
NO
2606
2607
2608
各係数の１ビットを周波数
ベースモデルでモデル化し
てエントロピー復号する
各係数の１ビットをＪＳ
Ｆモデルでモデル化して
エントロピー復号する
A

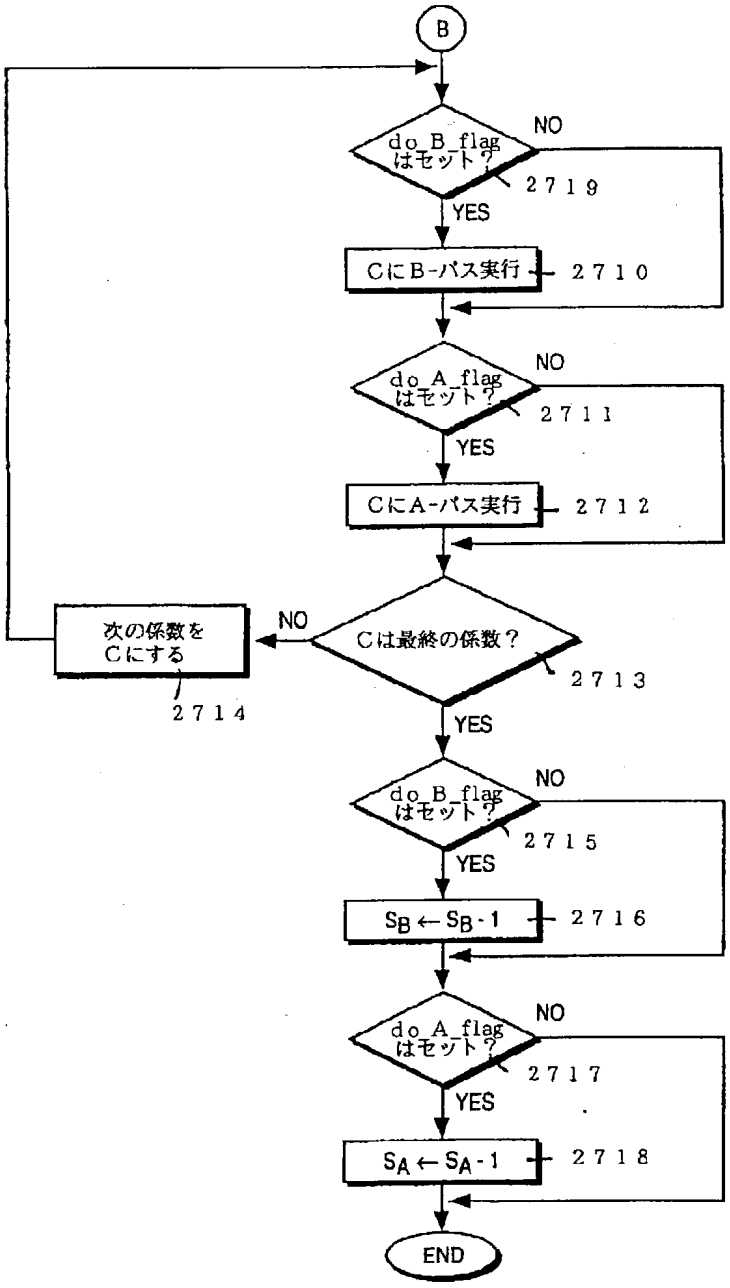
【図４７】


A
$S_A \geq 0$?
YES
B
2609
NO
$S_B \geq 0$?
YES
B
2610
NO
最も粗い分割レベルによる係数の逆 可逆フィルタをかける
2611
全レベルが逆フィルタされた？
NO
残っている最も粗い分割レベルによる係数に逆可逆フィルタをかける
2612
2613
YES
復元データを格納／伝送
2614
END

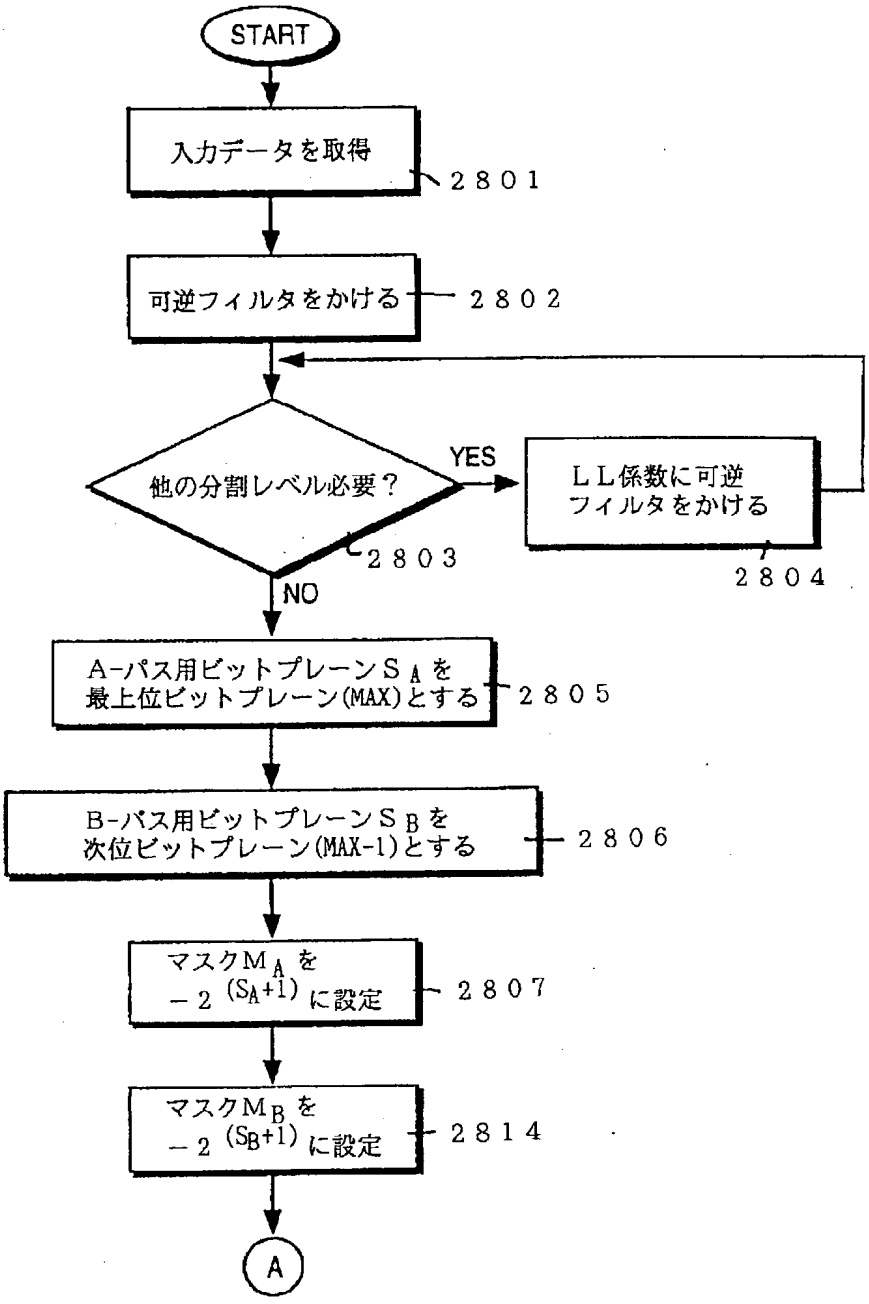
【図４８】


START
A
1パス？
NO
2701
YES
$S_A > S_B$ ？
NO
2702
YES
do_A_flag をクリア
2703
do_A_flag をセット
2704
$S_B = S_A$ ？
NO
2705
YES
do_B_flag をクリア
2705b
do_B_flag をセット
2706
do_A_flagは セット？ かつゼロ ツリーモデリング か？
NO
2707
YES
各係数の“確定／未確定”フラグを“未確定”に初期設定
2708
最初の係数を Cにする
2709
B

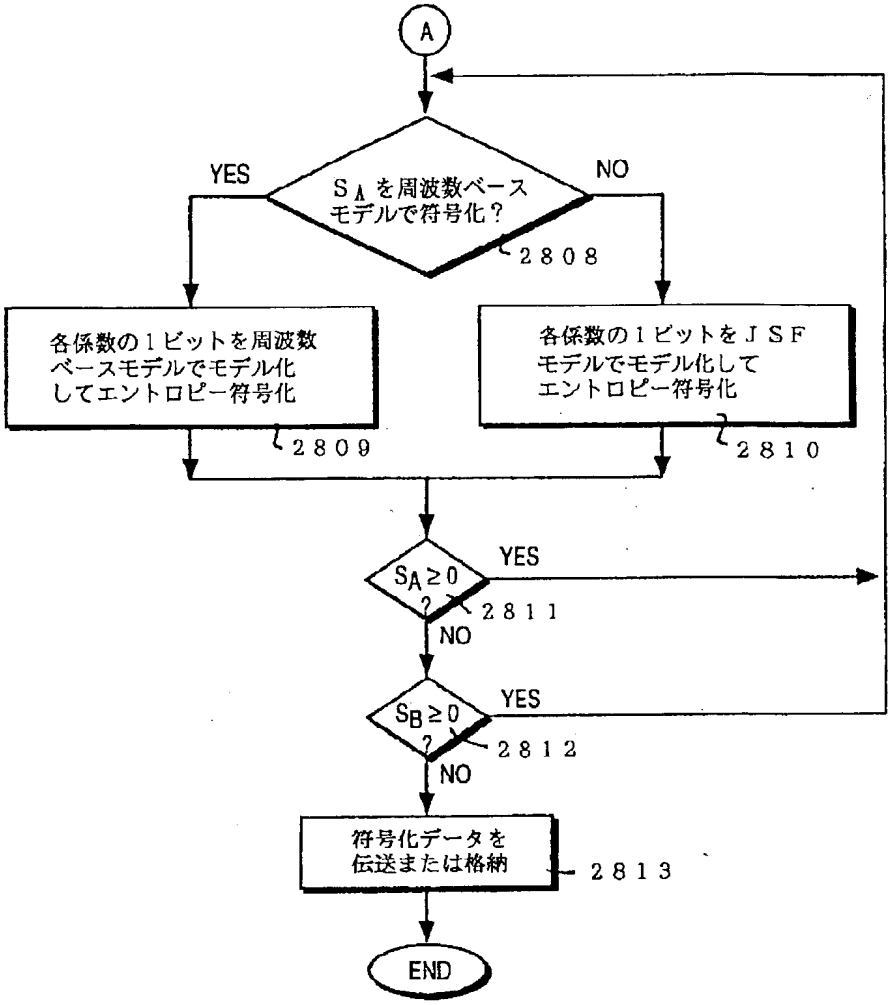
【図４９】


B
do_B_flag はセット？
NO
2719
YES
CにB-パス実行
2710
do_A_flag はセット？
NO
2711
YES
CにA-パス実行
2712
Cは最終の係数？
NO
次の係数を Cにする
2714
2713
YES
do_B_flag はセット？
NO
2715
YES
SB ← SB - 1
2716
do_A_flag はセット？
NO
2717
YES
SA ← SA - 1
2718
END

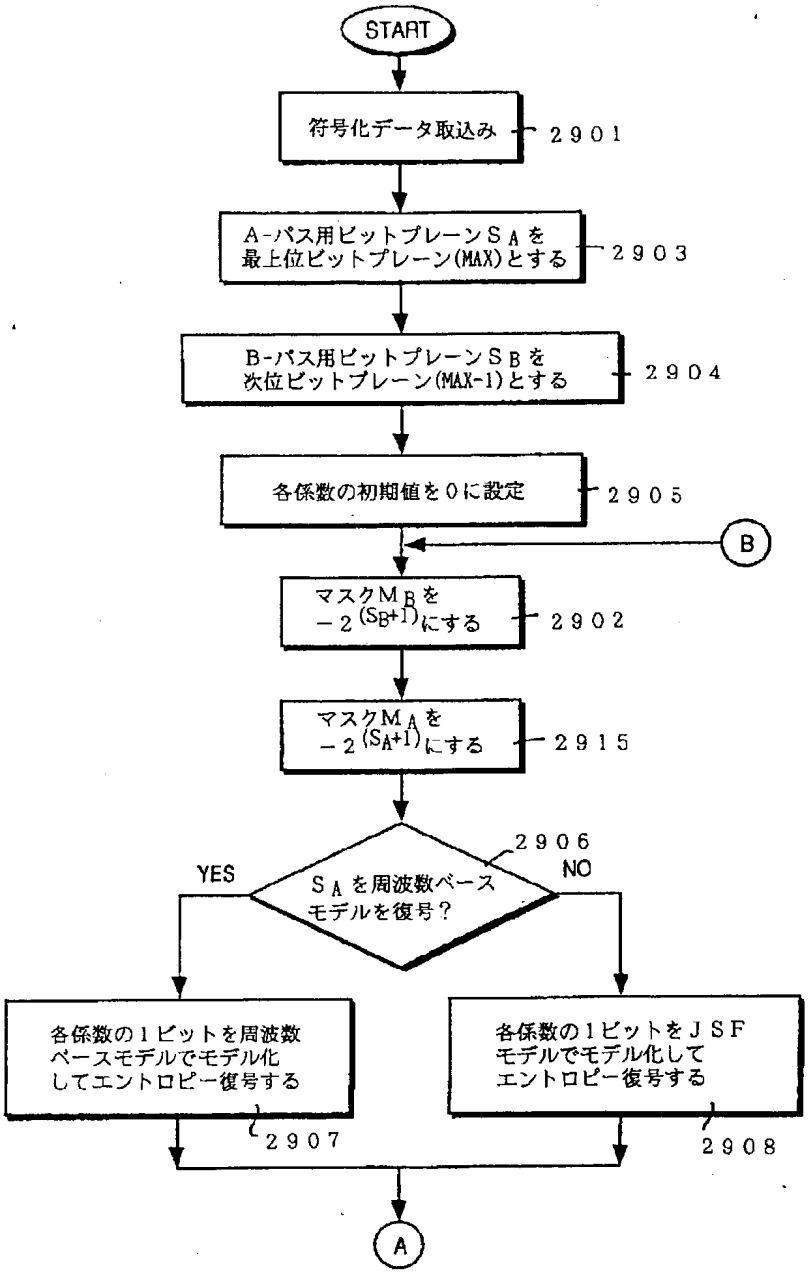
【図５０】


START
入力データを取得
２８０１
可逆フィルタをかける
２８０２
他の分割レベル必要？
YES
NO
２８０３
ＬＬ係数に可逆
フィルタをかける
２８０４
A-パス用ビットプレーンＳ A を
最上位ビットプレーン(MAX)とする
２８０５
B-パス用ビットプレーンＳ B を
次位ビットプレーン(MAX-1)とする
２８０６
マスクＭ A を
－２ (SA+1) に設定
２８０７
マスクＭ B を
－２ (SB+1) に設定
２８１４
A

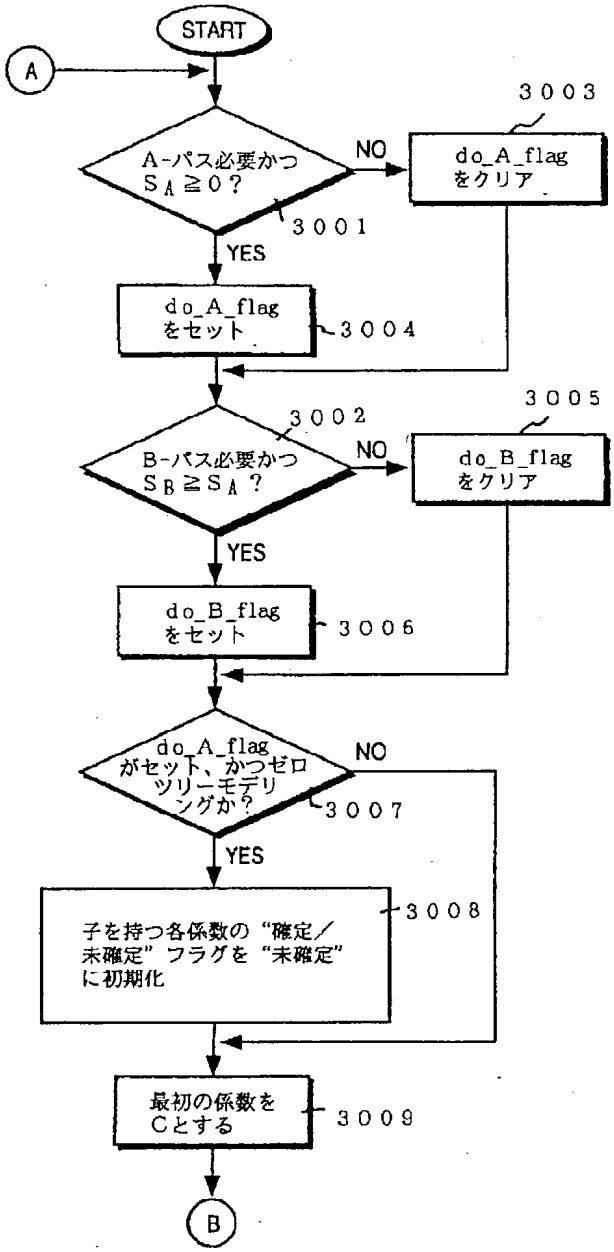
【図５１】


A
YES
S_Aを周波数ベース
モデルで符号化？
NO
2808
各係数の１ビットを周波数
ベースモデルでモデル化
してエントロピー符号化
2809
各係数の１ビットをＪＳＦ
モデルでモデル化して
エントロピー符号化
2810
$S_A \geq 0$
？
YES
2811
NO
$S_B \geq 0$
？
YES
2812
NO
符号化データを
伝送または格納
2813
END

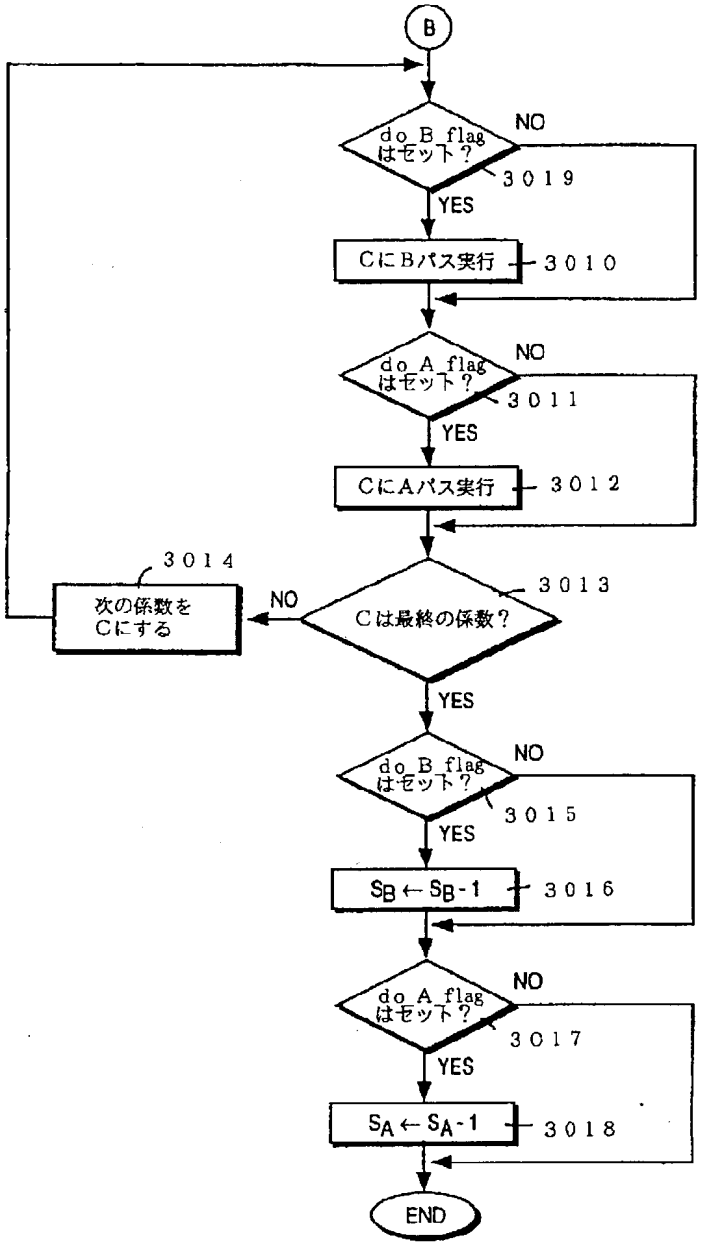
【図５２】


START
符号化データ取込み
2901
A-パス用ビットプレーンＳＡを
最上位ビットプレーン(MAX)とする
2903
B-パス用ビットプレーンＳＢを
次位ビットプレーン(MAX-1)とする
2904
各係数の初期値を０に設定
2905
B
マスクＭＢを
$-2^{(S_B+1)}$にする
2902
マスクＭＡを
$-2^{(S_A+1)}$にする
2915
2906
YES
ＳＡを周波数ベース
モデルを復号？
NO
各係数の１ビットを周波数
ベースモデルでモデル化
してエントロピー復号する
2907
各係数の１ビットをＪＳＦ
モデルでモデル化して
エントロピー復号する
2908
A

【図５４】


START
A
A-パス必要かつ
$S_A \geqq 0$?
3001
NO
3003
do_A_flag
をクリア
YES
do_A_flag
をセット
3004
3002
B-パス必要かつ
$S_B \geqq S_A$?
NO
3005
do_B_flag
をクリア
YES
do_B_flag
をセット
3006
do_A_flag
がセット、かつゼロ
ツリーモデリ
ングか?
3007
NO
YES
子を持つ各係数の“確定／
未確定”フラグを“未確定”
に初期化
3008
最初の係数を
Cとする
3009
B

【図55】


B
do_B_flag はセット？
NO
3019
YES
CにBパス実行
3010
do_A_flag はセット？
NO
3011
YES
CにAパス実行
3012
3014
次の係数を Cにする
NO
C は最終の係数？
3013
YES
do_B_flag はセット？
NO
3015
YES
SB ← SB - 1
3016
do_A_flag はセット？
NO
3017
YES
SA ← SA - 1
3018
END

【図５７】

ＲＴＳ変換（フォワード）－１レベル分割

```
void RTS(T,w,m,n)
int w,m,n;
int *T;
{
  int i,j,k;
  int *X, *Y;

  for (i=0;i<n;i++)
  {
    for (j=0;j<m;j+=2)
    {
      X[j/2] = (T[i*w+j]+T[i*w+j+1])>>1;
      X[j/2] = T[i*w+j]-T[i*w+j+1];
    }
    for (j=0;j<m/2;j++)
    {
      T[i*m+j]=X[j];
      T[i*m+m/2+j]= (-X[j]+(X[(j+1)%(m/2)]<<2)+X[(j+2)%(m/2)])>>2;
    }
  }
```

【図５８】

ＲＴＳ変換（インバース）－１レベル分割

```
void InverseRTS(T,w,m,n)
int w,m,n;
int *T;
{
  int i,j;
  int  *B, *D;

  for (i=0;i<n;i++)
  {
    for (j=0;j<m/2;j++)
    {
      B[j] = (-T[i*m+j]+T[i*m+((j+2)%(m/2))])&0X3;
      D[(j+1)%(m/2)] = ((T[i*m+j]-T[i*m+((j+2)%(m/2))]+
                        (T[i*m+m/2+j]<<2)+B[j])>>2);
    }
    for (j=0;j<m/2;j++)
    {
        T[i*w+2*j] = ((T[i*m+j]<<1)+D[j]+1)>>1;
        T[i*w+2*j+1] = T[i*w+2*j] - D[j];
    }
  }
}
```

【図５９】

ＲＴＳ変換（フォワード）－２レベル分割

```
void RTS(T,w,m,n)
int w,m,n;
int *T;
{
  int i,j,k;
  int *temp;
  int *X, *Y;

  for (i=0;i<n;i++)
  {
    for (j=0;j<m;j+=2)
    {
      X[j/2] = (T[i*w+j]+T[i*w+j+1])>>1;
      Y[j/2] = T[i*w+j]-T[i*w+j+1];
    }
    for (j=0;j<m/2;j++)
    {
      temp[i*m+j]=X[j];
      temp[i*m+m/2+j]= (-X[j]+(Y[(j+1)%(m/2)]<<2)+X[(j+2)%(m/2)])>>2;
    }
  }

  for (j=0;j<m;j++)
  {
    for (i=0;i<n;i+=2)
    {
      X[i/2] = (temp[i*m+j]+temp[(i+1)*m+j])>>1;
      Y[i/2] = temp[i*m+j]-temp[(i+1)*m+j];
    }
    for (i=0;i<n/2;i++)
    {
      T[i*w+j]=X[i];
      T[(i+n/2)*w+j]= (-X[i]+(Y[(i+1)%(n/2)]<<2)+X[(i+2)%(n/2)])>>2;
    }
  }
}
```

【図６０】

ＲＴＳ変換（インバース）－２レベル分割

```
void InverseRTS(T,w,m,n)
int w,m,n;
int *T;
{
  int i,j;
  int  *B, *D;
  int *temp;

  for (j=0;j<m;j++)
  {
    for (i=0;i<n/2;i++)
    {
      B[i] = (-T[i*w+j]+T[((i+2)%(n/2))*w+j])&0X3;
      D[(i+1)%(n/2)] = ((T[i*w+j]-T[((i+2)%(n/2))*w+j]+(T[(n/2+i)*w+j]<<2)+B[i])>>2);
    }
    for (i=0;i<n/2;i++)
    {
        temp[2*i*m+j] = ((T[i*w+j]<<1)+D[i]+1)>>1;
        temp[(2*i+1)*m+j] = temp[2*i*m+j] - D[i];
    }
  }

  for (i=0;i<n;i++)
  {
    for (j=0;j<m/2;j++)
    {
      B[j] = (-temp[i*m+j]+temp[i*m+((j+2)%(m/2))])&0X3;
      D[(j+1)%(m/2)] = ((temp[i*m+j]-temp[i*m+((j+2)%(m/2))]+
                        (temp[i*m+m/2+j]<<2)+B[j])>>2);
    }
    for (j=0;j<m/2;j++)
    {
        T[i*w+2*j] = ((temp[i*m+j]<<1)+D[j]+1)>>1;
        T[i*w+2*j+1] = T[i*w+2*j] - D[j];
    }
  }
}
```

フロントページの続き

(72)発明者　エドワード　エル　シュワル
アメリカ合衆国　カリフォルニア州
94025　メンローパーク　サンド　ヒル
ロード　2882　リコーコーポレーション内

(72)発明者　マーティン　ピー　ボーリック
アメリカ合衆国　カリフォルニア州
94025　メンローパーク　サンド　ヒル
ロード　2882　リコーコーポレーション内